Embedding Methods for NLP

Part 1: Unsupervised and Supervised Embeddings

Jason Weston & Antoine Bordes
Facebook AI Research

EMNLP tutorial – October 29, 2014

What is a word embedding?

Suppose you have a dictionary of words.

The i^{th} word in the dictionary is represented by an embedding:

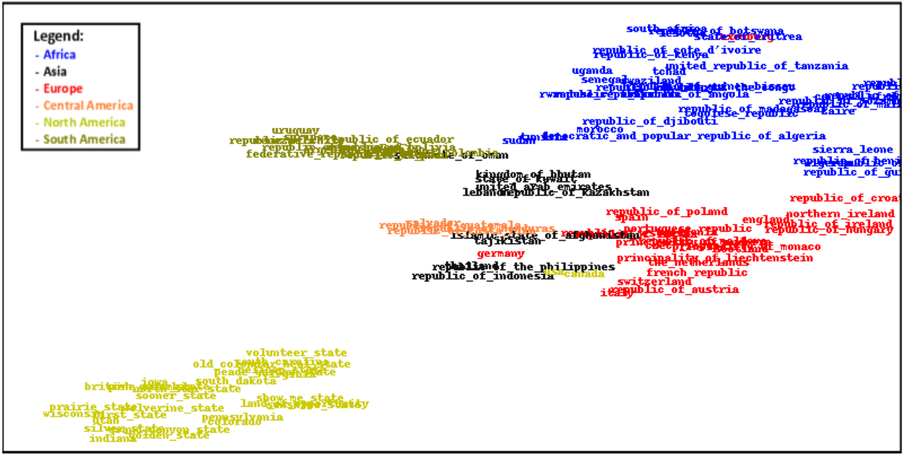
$$w_i \in \mathbb{R}^d$$

i.e. a d-dimensional vector, which is **learnt!**

- d typically in the range 50 to 1000.
- Similar words should have similar embeddings (share latent features).
- Embeddings can also be applied to *symbols* as well as words (e.g. Freebase nodes and edges).
- Discuss later: can also have embeddings of phrases, sentences, documents, or even other modalities such as images.

Learning an Embedding Space

Example of Embedding of 115 Countries (Bordes et al., '11)



Main methods we highlight, ordered by date.

- Latent Semantic Indexing (Deerwester et al., '88).
- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Convolutional Nets for tagging (SENNA) (Collobert & Weston, '08).
- Supervised Semantic Indexing (Bai et al, '09).
- Wsabie (Weston et al., '10).
- Recurrent NN-LMs (Mikolov et al., '10).
- Recursive NNs (Socher et al., '11).
- Word2Vec (Mikolov et al., '13).
- Paragraph Vector (Le & Mikolov, '14).
- Overview of recent applications.

Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:

- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al, '09).
- Wsabie (Weston et al., '10).

Embeddings for Language Modeling (useful for speech, translation, . . .):

- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Recurrent NN-LMs (Mikolov et al., '10).
- Word2Vec (Mikolov et al., '13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):

- Convolutional Nets for tagging (SENNA) (Collobert & Weston, '08).
- Recursive NNs (Socher et al., '11).
- Paragraph Vector (Le & Mikolov, '14).

Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:

- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al, '09).
- Wsabie (Weston et al., '10).

Embeddings for Language Modeling (useful for speech, translation, . . .):

- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Recurrent NN-LMs (Mikolov et al., '10).
- Word2Vec (Mikolov et al., '13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):

- Convolutional Nets for tagging (SENNA) (Collobert & Weston, '08).
- Recursive NNs (Socher et al., '11).
- Paragraph Vector (Le & Mikolov, '14).

Ranking and Retrieval: The Goal

We want to learn to match a query (text) to a target (text).

Many classical supervised ranking methods use hand-coded features.

Methods like LSI that learn from words are unsupervised.

Supervised Semantic Indexing (SSI) uses supervised learning from text only:

Bai et al, Learning to Rank with (a Lot of) Word Features. Journal of Information Retrieval, '09.

Outperforms existing methods (on words) like TFIDF, LSI or a (supervised) margin ranking perceptron baseline.

Basic Bag-O'-Words

Bag-of-words + cosine similarity:

- Each doc. $\{d_t\}_{t=1}^{N} \subset \mathbb{R}^{\mathcal{D}}$ is a *normalized* bag-of-words.
- Similarity with query q is: $f(q, d) = q^\top d$

Doesn't deal with synonyms: bag vectors can be orthogonal

No machine learning at all

Latent semantic indexing (LSI)

Learn a linear embedding $\phi(d_i) = Ud_i$ via a reconstruction objective.

- Rank with: $f(q, d) \quad = \quad q^\top U^\top Ud \quad = \quad \phi(q)^\top \phi(d_i)$ [1].

    Uses "synonyms": *low-dimensional latent "concepts"*.

    Unsupervised machine learning: useful for goal?

[1] $f(q, d) = q^\top (U^\top U + \alpha I)d$ gives better results.
Also, usually normalize this $\rightarrow$ cosine similarity.

Supervised Semantic Indexing (*SSI*)

- Basic model: rank with

$$f(q,d) \;=\; q^\top W d = \;\sum_{i,j=1}^{\mathcal{D}} q_i \; W_{ij} \; d_j$$

  i.e. learn weights of polynomial terms between documents.
- Learn $W \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ (huge!) with click-through data or other labels.

Uses "synonyms"

Supervised machine learning: targeted for goal

Too Big/Slow?! Solution = Constrain W :

low rank → *embedding model!*

SSI: why is this a good model?

Classical bag-of-words doesnt work when there are few matching terms:
q=(kitten, vet, nyc)
d=(cat, veterinarian, new, york)

- Method $q^\top Wd$ learns that e.g. kitten and cat are highly related.
- E.g. if i is the index of kitten and j is the index of cat, then $W_{ij} > 0$ after training.

SSI: Why the Basic Model Sucks

- W is big : 3.4Gb if $\mathcal{D} = 30000$, 14.5Tb if $\mathcal{D} = 2.5M$.
- Slow: $q^\top W d$ computation has mn computations $q_j W_{ij} d_i$, where q and d have m and n nonzero terms.
- Or one computes $v = q^\top W$ once, and then vd for each document. Classical speed where query has $\mathcal{D}$ terms, assuming W is dense $\rightarrow$ still slow.
- One could minimize $||W||_1$ and attempt to make W sparse. Then at most mp times slower than classical model (with p nonzeros in a column.)

SSI Improved model: Low Rank W

Constrain W:

$$W = U^\top V + I.$$

U and V are $N \times \mathcal{D}$ matrices $\rightarrow$ smaller

Low dimensional "latent concept" space like LSI (same speed).

Differences: supervised, asymmetric, learns with I.

Variants:

- $W = I$: bag-of-words again.
- $W = D$, reweighted bag-of-words related to [Grangier and Bengio, 2005].
- $W = U^\top U + I$: symmetric.

SSI: Training via maximizing AUC

- Given a set of tuples $\mathcal{R}$ with a query q, a related document d^+ and an unrelated (or lower ranked) document d^-.
- We would like $f(q, d^+) > f(q, d^-)$.
- Minimize margin ranking loss [Herbrich et al., 2000]:

$$\sum_{(q,d^+,d^-)\in\mathcal{R}} \max(0, 1 - f(q, d^+) + f(q, d^-)).$$

Learning Algorithm Stochastic Gradient Descent: Fast & scalable.

Iterate	Sample a triplet (q, d^+, d^-), Update $W \leftarrow W - \lambda \frac{\partial}{\partial W} \max(0, 1 - f(q, d^+) + f(q, d^-))$.

Other options: batch gradient, parallel SGD (hogwild), Adagrad ...

Training: setting hyperparameters

The following hyperparameters can be tuned for training:

- The initial random weights of the embedding vectors:
  e.g. use (mean 0, variance $\frac{1}{\sqrt{d}}$) .
- The learning rate (typically: 0.0001, 0.001, 0.01, 0.1, . . .).
- The value of the margin (e.g.: 1, 0.5, 0.2, 0.1, . . .).
- Restricting the norm of embeddings:
  $||U_i|| \leq C$, $||V_i|| \leq C$ (e.g.: C=1).

All these parameters are relative to each other, e.g. a larger margin might need larger initial weights and learning rate.
Typically, we fix the initialization and norm, and try different values of margin and learning rate. This can make big differences in performance.

Prior Work: Summary of learning to Rank

- [Grangier & Bengio, '06] used similar methods to basic SSI for retrieving images.
- [Goel, Langord & Strehl, '08] used Hash Kernels (Vowpal Wabbit) for advert placement.
- Main difference: SSI uses low rank on W.
- SVM [Joachims, 2002] and NN ranking methods [Burges, 2005] .
  Use hand-coded features: title, body, URL, search rankings,... (don't use words)
  (e.g. Burges uses 569 features in all).
- In contrast SSI uses only the words and trains on huge feature sets.
- Several works on optimizing different loss functions (MAP, ROC, NDCG): [Cao, 2008], [Yu, 2007], [Qin, 2006],....
- Lots of stuff for "metric learning" problem as well..

One could also add features + new loss to this method ..

Experimental Comparison

- **Wikipedia**
  - 1,828,645 documents. 24,667,286 links.
  - Split into 70% train, 30% test.
- Pick random doc. as query, then rank other docs.
- Docs that are linked to it should be highly ranked.
- **Two setups:**
  (i) whole document is used as query;
  (ii) 5,10 or 20 words are picked to mimic keyword search.

Wikipedia Experiments: Document Retrieval Performance

Experiments on Wikipedia, which contains 1.8M documents: retrieval task using the link structure and separated the data into 70% for training and 30% for test.

Document based retrieval:

Algorithm	Rank-Loss	MAP	P10
TFIDF	0.842%	0.432±0.012	0.193
αLSI + $(1-\alpha)$TFIDF	0.721%	0.433	0.193
Linear SVM Ranker	0.410%	0.477	0.212
Hash Kernels + αI	0.322%	0.492	0.215
SSI	0.158%	0.547±0.012	0.239±0.008

k-keywords based retrieval:

$k=5$:	Algorithm	Params	Rank	MAP	P@10
	TFIDF	0	21.6%	0.047	0.023
	αLSI + $(1-\alpha)$TFIDF	$200\mathcal{D}+1$	14.2%	0.049	0.023
	SSI	$400\mathcal{D}$	**4.37%**	**0.166**	**0.083**

Scatter Plots: SSI vs. TFIDF and LSI

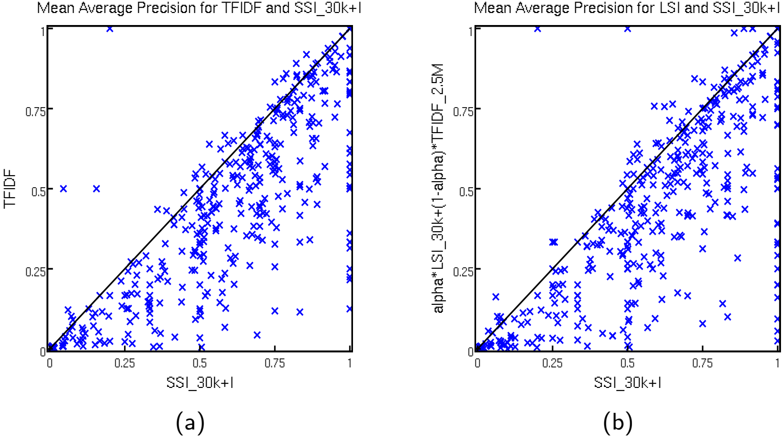


Figure : Scatter plots of Average Precision for 500 documents: (a) SSI vs. TFIDF, (b) SSI vs. αLSI + $(1-\alpha)$ TFIDF.

Experiments: Cross-Language Retrieval

Retrieval experiments using a query document in japanese, where the task is to retrieve documents in English (using link structure as ground truth).

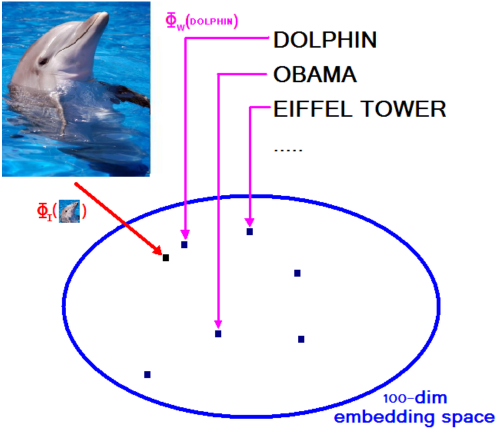
SSI can do this without doing a translation step first as it learns to map the two languages together in the embedding space.

Algorithm	Rank-Loss	MAP	P10
TFIDF_{EngEng}(Google translated queries)	4.78%	0.319	0.259
$\alpha\text{LSI}_{EngEng}+(1-\alpha)\text{TFIDF}_{EngEng}$	3.71%	0.300	0.253
$\alpha\text{CL-LSI}_{JapEng}+(1-\alpha)\text{TFIDF}_{EngEng}$	3.31%	0.275	0.212
SSI_{EngEng}	1.72%	0.399	0.325
SSI_{JapEng}	0.96%	0.438	0.351
$\alpha\text{SSI}_{JapEng} + (1-\alpha)\text{TFIDF}_{EngEng}$	0.75%	0.493	0.377
$\alpha\text{SSI}_{JapEng} + (1-\alpha)\text{SSI}_{EngEng}$	**0.63%**	**0.524**	**0.386**

Some recent related translation-based embeddings:
(Hermann & Blunsom, ICLR '14) and (Mikolov et al., '13).

Wsabie (Weston, Bengio & Usunier, '10)

- Extension to SSI, also embeds objects other than text, e.g. images.
- WARP loss function that optimizes precision@k.



Learn $\Phi_I(\cdot)$ and $\Phi_W(\cdot)$ to optimize precision@k.

Joint Item-Item Embedding Model

L.H.S: Image, query string or user profile (depending on the task)

$$\Phi_{LHS}(x) = U\Phi_x(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{100}.$$

R.H.S: document, image, video or annotation (depending on the task)

$$\Phi_{RHS}(y) = V\Phi_y(y) : \mathbb{R}^{d_y} \to \mathbb{R}^{100}.$$

This model again compares the degree of match between the L.H.S and R.H.S in the embedding space:

$$f_y(x) = sim(\Phi_{LHS}(x), \Phi_{RHS}(y)) = \Phi_x(x)^\top U^\top V\Phi_y(y)$$

Also constrain the weights (regularize):

$$||U_i||_2 \leq C, \quad i = 1, \ldots, d_x, \qquad ||V_i||_2 \leq C, \quad i = 1, \ldots, d_y.$$

Ranking Annotations: AUC is Suboptimal

Classical approach to learning to rank is maximize AUC by minimizing:

$$\sum_{x} \sum_{y} \sum_{\bar{y} \neq y} \max(0, 1 + f_{\bar{y}}(x) - f_{y}(x))$$

Problem: All pairwise errors are considered the same, it counts the number of ranking violations.

Example:

Function 1: true annotations ranked 1st and 101st.

Function 2: true annotations ranked 50th and 52nd.

AUC prefers these *equally* as both have 100 "violations".

We want to optimize the top of the ranked list!

Rank Weighted Loss [Usunier et al. '09]

Replace classical AUC optimization:

$$\sum_x \sum_y \sum_{\bar{y} \neq y} \max(0, 1 + f_{\bar{y}}(x) - f_y(x))$$

With weighted version:

$$\sum_x \sum_y \sum_{\bar{y} \neq y} L(rank_y(x)) \max(0, 1 + f_{\bar{y}}(x) - f_y(x))$$

where $rank_y(f(x))$ is the rank of the true label:

$$rank_y(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) \geq f_y(x))$$

and $L(\eta)$ converts the rank to a weight, e.g. $L(\eta) = \sum_{i=1}^{\eta} 1/\eta$.

Weighted Approximate-Rank Pairwise (WARP) Loss

Problem: we would like to apply SGD:

$$\textit{Weighting}\ L(rank_y(f(x))), \qquad rank_y(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) + 1 \geq f_y(x))$$

... too expensive to compute per (x, y) sample as $y \in \mathcal{Y}$ is large.

Solution: approximate by sampling $f_i(x)$ until we find a violating label $\bar{y}$

$$rank_y(f(x)) \approx \left\lfloor \frac{|\mathcal{Y}| - 1}{N} \right\rfloor$$

where N is the number of trials in the sampling step.

Online WARP Loss

Input: labeled data (x_i, y_i), $y_i \in \{1, \dots, Y\}$.
repeat
  Pick a random labeled example (x_i, y_i)
  Set $N = 0$.
  repeat
    Pick a random annotation $\bar{y} \in \{1, \dots, Y\} \setminus y_i$.
    $N = N + 1$.
  until $f_{\bar{y}}(x) > f_{y_i}(x) - 1$ or $N > Y - 1$
  if $f_{\bar{y}}(x) > f_{y_i}(x) - 1$ **then**
    Make a gradient step to minimize:

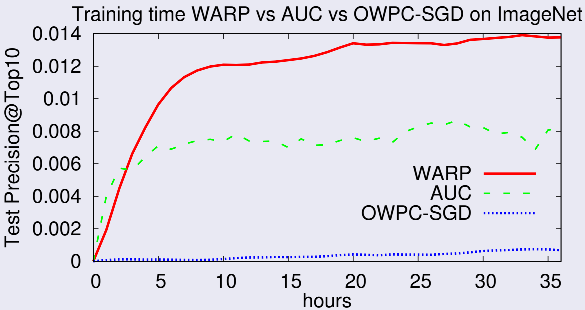
$$L(\lfloor \frac{Y-1}{N} \rfloor)|1 - f_y(x) + f_{\bar{y}}(x)|_+$$

  end if
until validation error does not improve.

Image Annotation Performance

Algorithm	16k ImageNet	22k ImageNet	97k Web Data
Nearest Means	4.4%	2.7%	2.3%
One-vs-all SVMs 1+:1-	4.1%	3.5%	1.6%
One-vs-all SVMs	9.4%	8.2%	6.8%
AUC SVM Ranker	4.7%	5.1%	3.1%
Wsabie	11.9%	10.5%	8.3%

Training time: WARP vs. OWPC-SGD & AUC



Learned Annotation Embedding (on Web Data)

Annotation	Neighboring Annotations
barack obama	*barak obama*, obama, barack, barrack obama, bow wow
david beckham	*beckham*, *david beckam*, alessandro del piero, *del piero*
santa	*santa claus*, *papa noel*, *pere noel*, santa clause, joyeux noel
dolphin	delphin, dauphin, *whale*, *delfin*, *delfini*, *baleine*, blue whale
cows	*cattle*, *shire*, dairy cows, kuh, *horse*, cow, *shire horse*, *kone*
rose	rosen, *hibiscus*, *rose flower*, rosa, roze, pink rose, *red rose*
pine tree	*abies alba*, abies, *araucaria*, pine, neem tree, *oak tree*
mount fuji	mt fuji, fuji, fujisan, fujiyama, *mountain*, *zugspitze*
eiffel tower	*eiffel*, tour eiffel, la tour eiffel, *big ben*, *paris*, blue mosque
ipod	i pod, *ipod nano*, apple ipod, ipod apple, new ipod
f18	f 18, eurofighter, f14, fighter jet, tomcat, mig 21, f 16

Summary

Conclusion

- Powerful: supervised methods for ranking.
  - Outperform classical methods
- Efficient low-rank models $\rightarrow$ learn hidden representations.
- Embeddings good for generalization, but can "blur" too much e.g. for exact word matches.

Extensions

- Nonlinear extensions – e.g. convolutional net instead.
  e.g. DeViSE (Frome et al., NIPS '13)

Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:

- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al, '09).
- Wsabie (Weston et al., '10).

Embeddings for Language Modeling (useful for speech, translation)

- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Recurrent NN-LMs (Mikolov et al., '10).
- Word2Vec (Mikolov et al., '13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):

- Convolutional Nets for tagging (SENNA) (Collobert & Weston, '08).
- Recursive NNs (Socher et al., '11).
- Paragraph Vector (Le & Mikolov, '14).

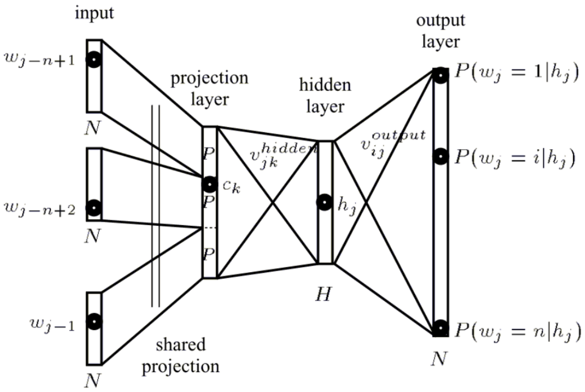
Language Modeling

Task: given a sequence of words, predict the next word.

the cat sat on the **??**

- *n*-gram models are a strong baseline on this task.
- A variety of embedding models have been tried, they can improve results.
- The embeddings learnt from this unsupervised task can also be used to transfer to and improve a supervised task.

Neural Network Language Models



Bengio, Y., Schwenk, H., Sencal, J. S., Morin, F., & Gauvain, J. L. (2006). Neural probabilistic language models. In Innovations in Machine Learning (pp. 137-186). Springer Berlin Heidelberg.

Neural Network Language Models: Hierarchical Soft Max Trick (Morin & Bengio '05)

Predicting the probability of each next word is slow in NNLMs because the output layer of the network is the size of the dictionary.

Can predict via a tree instead:

1. Cluster the dictionary either according to semantics (similar words in the same cluster) or frequency (common words in the same cluster). *This gives a two-layer tree, but a binary tree is another possibility.*
2. The internal nodes explicitly model the probability of its child nodes.
3. The cost of predicting the probability of the true word is now: traversal to the child, plus normalization via the internal nodes and children in the same node.

This idea is used in Word2Vec and RNN models as well.

Recurrent Neural Network Language Models

Key idea: *input to predict next word is current word plus context fed-back from previous word (i.e. remembers the past with recurrent connection).*

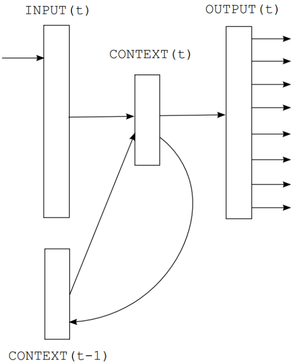


Figure: *Recurrent neural network based LM*

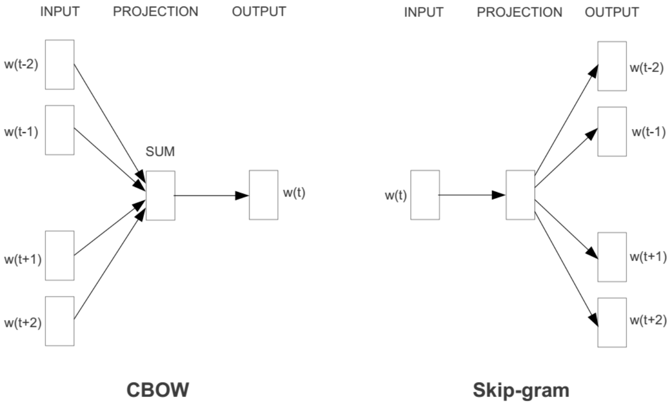
Recurrent neural network based language model. Mikolov et al., Interspeech, '10.

NNLMS vs. RNNS: Penn Treebank Results (Mikolov)

Model	Weight	PPL
3-gram with Good-Turing smoothing (GT3)	0	165.2
5-gram with Kneser-Ney smoothing (KN5)	0	141.2
5-gram with Kneser-Ney smoothing + cache	0.0792	125.7
Maximum entropy model	0	142.1
Random clusterings LM	0	170.1
Random forest LM	0.1057	131.9
Structured LM	0.0196	146.1
Within and across sentence boundary LM	0.0838	116.6
Log-bilinear LM	0	144.5
Feedforward NNLM	0	140.2
Syntactical NNLM	0.0828	131.3
Combination of static RNNLMs	0.3231	102.1
Combination of adaptive RNNLMs	0.3058	101.0
ALL	1	**83.5**

Recent uses of NNLMs and RNNs to improve machine translation:
Fast and Robust NN Joint Models for Machine Translation, Devlin et al, ACL '14.
Also (Kalchbrenner '13), (Sutskever et al., '14), (Cho et al., '14).

Word2Vec : very simple LM, works well



Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. NIPS, 2013.

Word2Vec: compositionality

Table 7: *Comparison and combination of models on the Microsoft Sentence Completion Challenge.*

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Code: `https://code.google.com/p/word2vec/`

Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:

- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al, '09).
- Wsabie (Weston et al., '10).

Embeddings for Language Modeling (useful for speech, translation, . . .):

- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Recurrent NN-LMs (Mikolov et al., '10).
- Word2Vec (Mikolov et al., '13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):

- Convolutional Nets for tagging (SENNA) (Collobert & Weston, '08).
- Recursive NNs (Socher et al., '11).
- Paragraph Vector (Le & Mikolov, '14).

NLP Tasks

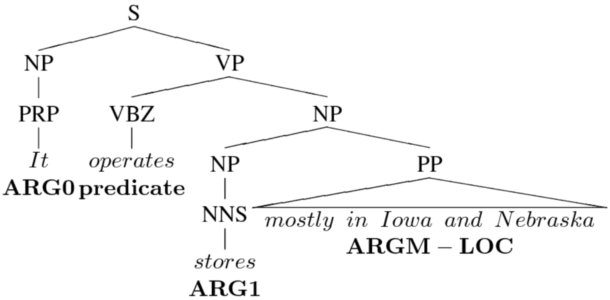
- Part-Of-Speech Tagging (POS): syntactic roles (noun, adverb...)
- Chunking: syntactic constituents (noun phrase, verb phrase...)
- Name Entity Recognition (NER): person/company/location...
- Semantic Role Labeling (SRL):

  [John]$_{ARG0}$ [ate]$_{REL}$ [the apple]$_{ARG1}$ [in the garden]$_{ARGM-LOC}$

Complex Systems

- Two extreme choices to get a complex system
  - ⋆ Large Scale Engineering: design a lot of complex features, use a fast existing linear machine learning algorithm
  - ⋆ Large Scale Machine Learning: use simple features, design a complex model which will implicitly learn the right features

The Large Scale Feature Engineering Way



- Extract **hand-made features** e.g. from the parse tree
- Disjoint: all tasks trained separately, Cascade features
- Feed these features to a shallow classifier like SVM

ASSERT: many hand built features for SRL (Pradhan et al, '04)

Problems:

1) Features rely on other solutions (parsing, named entity, word-sense)

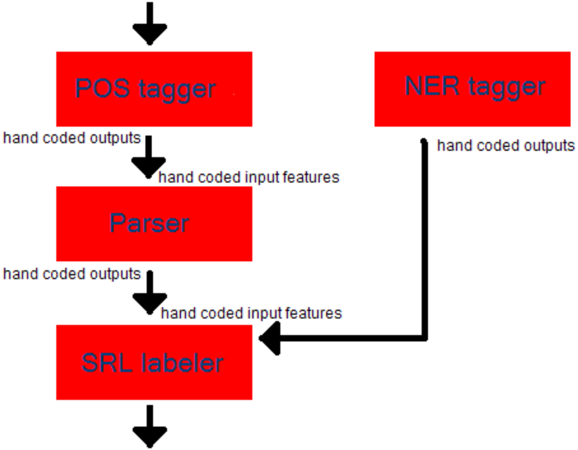
2) Technology task-transfer is difficult

- Choose some **good hand-crafted features**

Predicate and POS tag of predicate	Voice: active or passive (hand-built rules)
Phrase type: adverbial phrase, prepositional phrase, ...	Governing category: Parent node's phrase type(s)
Head word and POS tag of the head word	Position: left or right of verb
Path: traversal from predicate to constituent	Predicted named entity class
Word-sense disambiguation of the verb	Verb clustering
Length of the target constituent (number of words)	NEG feature: whether the verb chunk has a "not"
Partial Path: lowest common ancestor in path	Head word replacement in prepopositional phrases
First and last words and POS in constituents	Ordinal position from predicate + constituent type
Constituent tree distance	Temporal cue words (hand-built rules)
Dynamic class context: previous node labels	Constituent relative features: phrase type
Constituent relative features: head word	Constituent relative features: head word POS
Constituent relative features: siblings	Number of pirates existing in the world...

- Feed them to a **shallow classifier** like SVM

The Suboptimal (?) Cascade



(Or, the opposing view is the above is a smart use of prior knowledge..)

NLP: Large Scale Machine Learning

Goals

- Task-specific engineering limits NLP scope
- Can we find unified hidden representations?
- Can we build unified NLP architecture?

Means

- Start from scratch: forget (most of) NLP knowledge
- Compare against classical NLP benchmarks
- Our dogma: avoid task-specific engineering

NLP Benchmarks

- Datasets:
  - ⋆ POS, CHUNK, SRL: WSJ (≈ up to 1M labeled words)
  - ⋆ NER: Reuters (≈ 200K labeled words)

System	Accuracy
Shen, 2007	97.33%
Toutanova, 2003	**97.24%**
Gimenez, 2004	97.16%

(a) POS: As in (Toutanova, 2003)

System	F1
Shen, 2005	95.23%
Sha, 2003	**94.29%**
Kudoh, 2001	93.91%

(b) CHUNK: CoNLL 2000

System	F1
Ando, 2005	**89.31%**
Florian, 2003	88.76%
Kudoh, 2001	88.31%

(c) NER: CoNLL 2003

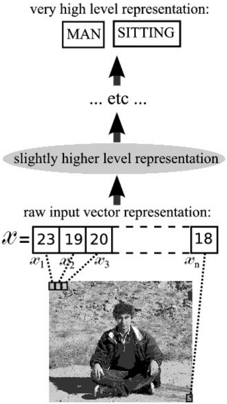
System	F1
Koomen, 2005	**77.92%**
Pradhan, 2005	77.30%
Haghighi, 2005	77.04%

(d) SRL: CoNLL 2005

- We chose as benchmark systems:
  - ⋆ Well-established systems
  - ⋆ Systems avoiding external labeled data
- Notes:
  - ⋆ Ando, 2005 uses external unlabeled data
  - ⋆ Koomen, 2005 uses 4 parse trees not provided by the challenge

The "Deep Learning" Way

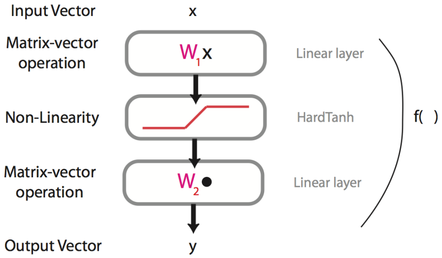
Neural nets attempt to propose a radically? different end-to-end approach:



- Avoid building a parse tree. Humans don't need this to talk.
- Try to avoid all hand-built features $\rightarrow$ monolithic systems.

Neural Networks

- Stack several layers together

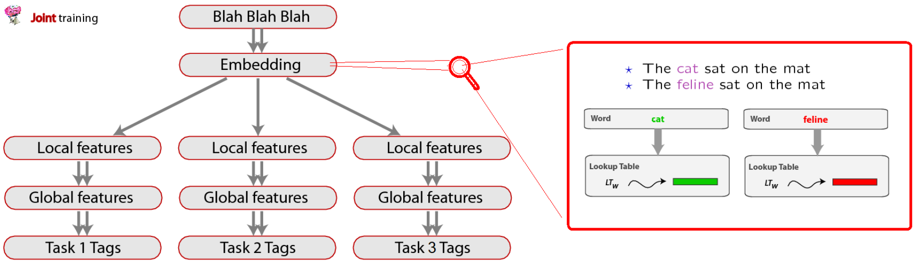


- Increasing level of abstraction at each layer
- Requires simpler features than "shallow" classifiers
- The "weights" W_i are trained by gradient descent
- How can we feed words?

The Big Picture

A unified architecture for all NLP (labeling) tasks:

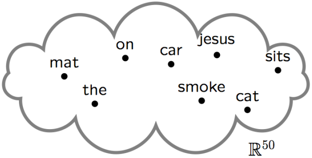
Sentence:	*Felix*	*sat*	*on*	*the*	*mat*	.
POS:	NNP	VBD	IN	DT	NN	.
CHUNK:	NP	VP	PP	NP	NP-I	.
NER:	PER	-	-	-	-	-
SRL:	ARG1	REL	ARG2	ARG2-I	ARG2-I	-



Words into Vectors

Idea

- Words are embed in a vector space



- Embeddings are trained

Implementation

- A word w is an index in a dictionary $\mathcal{D} \in \mathbb{N}$
- Use a lookup-table ($W \sim$ feature size $\times$ dictionary size)

$$LT_W(w) = W_{\bullet w}$$

Remarks

- Applicable to any discrete feature (words, caps, stems...)
- See (Bengio et al, 2001)

The Lookup Tables

Each word/element in dictionary maps to a vector in $\mathbb{R}^d$.

- We learn these vectors.
- LookupTable: input of i^{th} word is

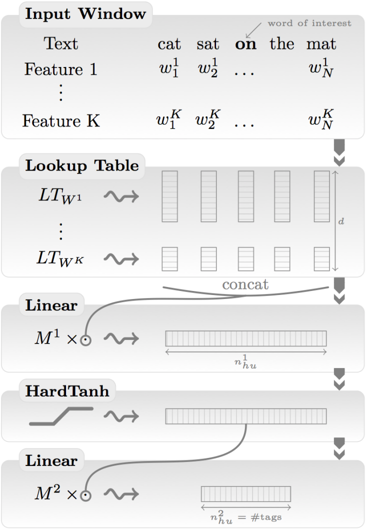
$$x = (0,\, 0,\, \ldots,\, 1,\, 0,\, \ldots,\, 0) \quad \text{1 at position } i$$

In the original space words are orthogonal.

cat = (0,0,0,0,0,0,0,0,0,1,0,0,0,0, ...)
kitten = (0,0,1,0,0,0,0,0,0,0,0,0,0,0, ...)

To get the $\mathbb{R}^d$ embedding vector for the word we multiply Wx where W is a $d \times N$ vector with N words in the dictionary.

Window Approach

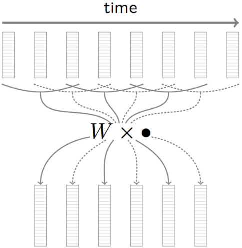


- Tags one word at the time
- Feed a fixed-size window of text around each word to tag
- Works fine for most tasks
- How do deal with long-range dependencies?

  E.g. in SRL, the verb of interest might be outside the window!

Sentence Approach (1/2)

- Feed the whole sentence to the network
- Tag one word at the time: add extra position features
- Convolutions to handle variable-length inputs

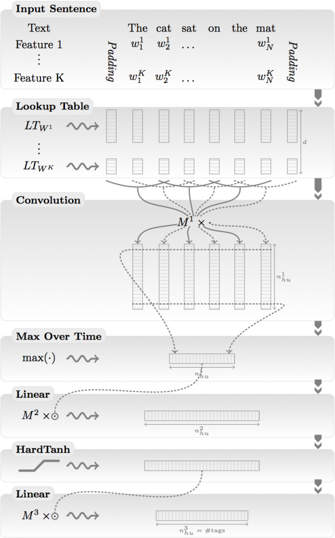


See (Bottou, 1989)
or (LeCun, 1989).

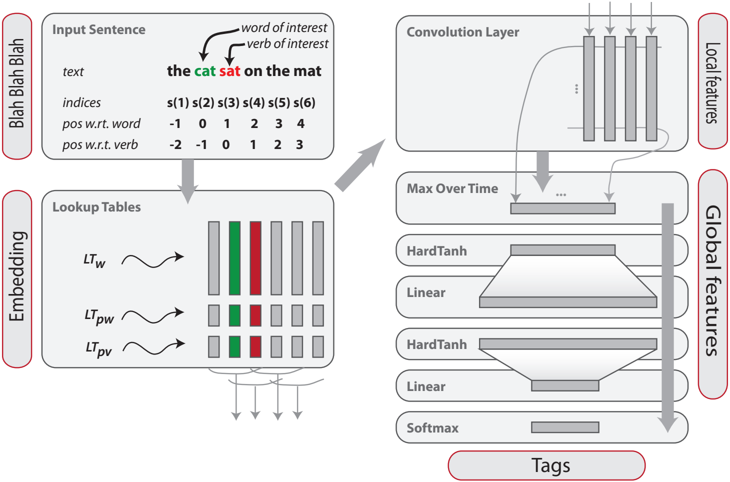
- Produces local features with higher level of abstraction
- Max over time to capture most relevant features

Outputs a fixed-sized feature vector

Sentence Approach (2/2)

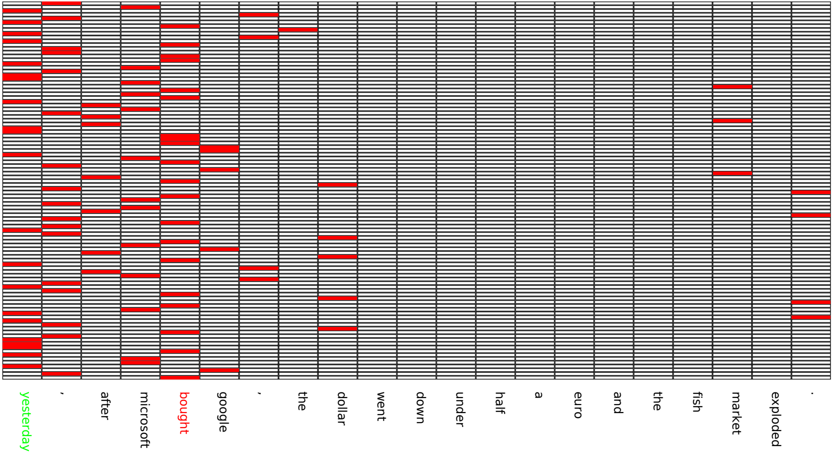


Deep SRL

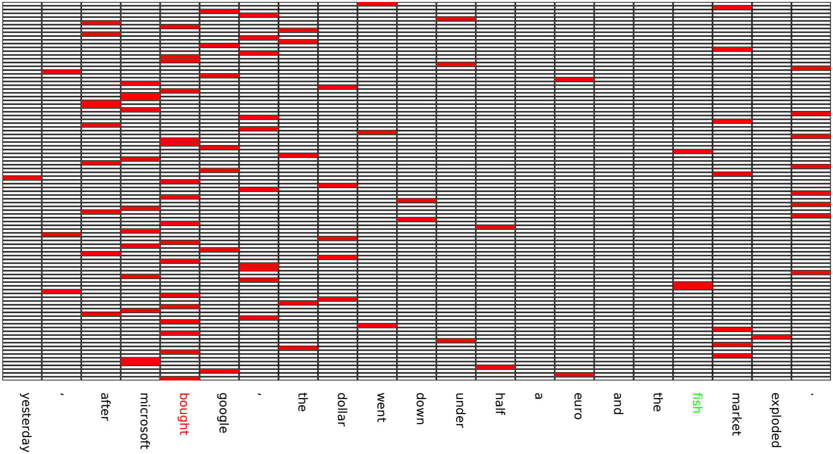


This is the network for a single window. We train/test predicting the entire sentence of tags ("structured outputs") using viterbi approach, similar to other NLP methods.

Removing The Time Dimension (1/2)



Removing The Time Dimension (2/2)



Word Tag Likelihood (WTL)

- The network has one output $f(\boldsymbol{x}, i, \boldsymbol{\theta})$ per tag i
- Interpreted as a probability with a softmax over all tags

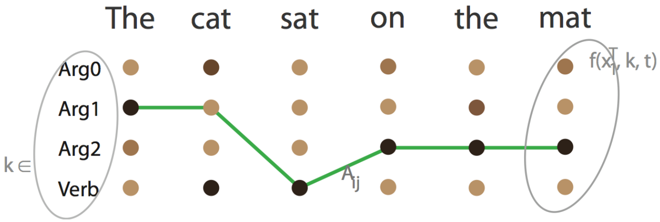
$$p(i \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{e^{f(\boldsymbol{x}, i, \boldsymbol{\theta})}}{\sum_j e^{f(\boldsymbol{x}, j, \boldsymbol{\theta})}}$$

.. we can train directly for that (word tag likelihood) or we could train in a structured way by predicting the entire sentence's tags.

That should be useful because tags are not independent.

Sentence Tag Likelihood (STL)

- The network score for tag k at the t^{th} word is $f([\boldsymbol{x}]_1^T,\, k,\, t,\, \boldsymbol{\theta})$
- A_{kl} transition score to jump from tag k to tag l



- Sentence score for a tag path $[i]_1^T$

$$s([\boldsymbol{x}]_1^T,\, [i]_1^T,\, \tilde{\boldsymbol{\theta}}) = \sum_{t=1}^{T} \left(A_{[i]_{t-1}[i]_t} + f([\boldsymbol{x}]_1^T,\, [i]_t,\, t,\, \boldsymbol{\theta}) \right)$$

Supervised Benchmark Results

- Network architectures:
  - ⋆ Window (5) approach for POS, CHUNK & NER (300HU)
  - ⋆ Convolutional (3) for SRL (300+500HU)
  - ⋆ Word Tag Likelihood (WTL) and Sentence Tag Likelihood (STL)
- Network features: lower case words (size 50), capital letters (size 5)
  dictionary size 100,000 words

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
Benchmark Systems	**97.24**	**94.29**	**89.31**	**77.92**
NN+WTL	96.31	89.13	79.53	55.40
NN+STL	96.37	90.33	81.47	*70.99*

- STL helps, but... fair performance.
- Capacity mainly in words features... are we training it right?

Supervised Word Embeddings

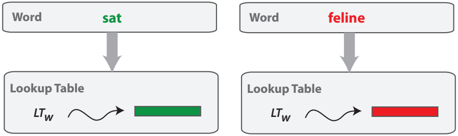
- Sentences with similar words should be tagged in the same way:
  - ⋆ The cat sat on the mat
  - ⋆ The feline sat on the mat

france	jesus	xbox	reddish	scratched	megabits
454	1973	6909	11724	29869	87025
persuade	thickets	decadent	widescreen	odd	ppa
faw	savary	divo	antica	anchieta	uddin
blackstock	sympathetic	verus	shabby	emigration	biologically
giorgi	jfk	oxide	awe	marking	kayak
shaheed	khwarazm	urbina	thud	heuer	mclarens
rumelia	stationery	epos	occupant	sambhaji	gladwin
planum	ilias	eglinton	revised	worshippers	centrally
goa'uld	gsNUMBER	edging	leavened	ritsuko	indonesia
collation	operator	frg	pandionidae	lifeless	moneo
bacha	w.j.	namsos	shirt	mahan	nilgiris

- About 1M of words in WSJ
- 15% of most frequent words in the dictionary are seen 90% of the time
- Cannot expect words to be trained properly!

Improving Word Embedding

- Rare words are not trained properly
- Sentences with similar words should be tagged in the same way:
  - The cat sat on the mat
  - The feline sat on the mat



Only 1M WSJ not enough – let's use lots of unsupervised data!

Semi-supervised: MTL with Unlabeled Text

- Language Model: "*is a sentence actually english or not?*"
  Implicitly captures: * syntax * semantics
- Bengio & Ducharme (2001) Probability of next word given previous words. Overcomplicated – we do not need probabilities here
- English sentence windows: Wikipedia ($\sim 631M$ words)
  Non-english sentence windows: middle word randomly replaced

  the champion federer wins wimbledon
  vs. the champion saucepan wins wimbledon
- Multi-class margin cost:

$$\sum_{s \in \mathcal{S}} \sum_{w \in \mathcal{D}} \max\left(0,\, 1 - f(s,\, w_s^\star) + f(s,\, w)\right)$$

$\mathcal{S}$: sentence windows $\mathcal{D}$: dictionary

$w_s^\star$: true middle word in s

$f(s,\, w)$: network score for sentence s and middle word w

Language Model: Embedding

Nearest neighbors in 100-dim. embedding space:

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869
SPAIN	CHRIST	PLAYSTATION	YELLOWISH	SMASHED
ITALY	GOD	DREAMCAST	GREENISH	RIPPED
RUSSIA	RESURRECTION	PSNUMBER	BROWNISH	BRUSHED
POLAND	PRAYER	SNES	BLUISH	HURLED
ENGLAND	YAHWEH	WII	CREAMY	GRABBED
DENMARK	JOSEPHUS	NES	WHITISH	TOSSED
GERMANY	MOSES	NINTENDO	BLACKISH	SQUEEZED
PORTUGAL	SIN	GAMECUBE	SILVERY	BLASTED
SWEDEN	HEAVEN	PSP	GREYISH	TANGLED
AUSTRIA	SALVATION	AMIGA	PALER	SLASHED

(Even fairly rare words are embedded well.)

Results

Algorithm	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Baselines	97.24	94.29	89.31	77.92
	[Toutanova '03]	[Sha '03]	[Ando '05]	[Koomen '05]
NN + WTL	96.31	89.13	79.53	55.40
NN + STL	96.37	90.33	81.47	70.99
NN + LM + STL	97.22	94.10	88.67	74.15
NN + ... + tricks	97.29	94.32	89.95	76.03
	[+suffix]	[+POS]	[+gazetteer]	[+Parse Trees]

NOTES:

– Didn't compare to benchmarks that used external labeled data.

– [Ando '05] uses external unlabeled data.

– [Koomen '05] uses 4 parse trees not provided by the challenge. Using only 1 tree it gets 74.76.

Software

Code for tagging with POS, NER, CHUNK, SRL + parse trees:
http://ml.nec-labs.com/senna/

System	RAM (Mb)	Time (s)
Toutanova, 2003	1100	1065
Shen, 2007	2200	833
SENNA	32	4

(a) POS

System	RAM (Mb)	Time (s)
Koomen, 2005	3400	6253
SENNA	124	52

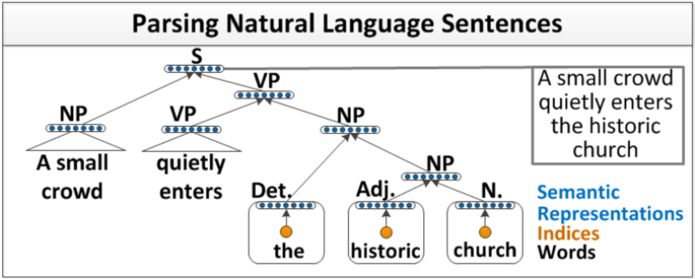
(b) SRL

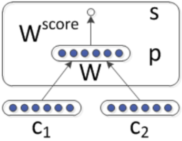
See also Torch: http://www.torch.ch

Recursive NNs for Parsing, Sentiment, ... and more!

(Socher et al., ICML '13), (Socher et al., EMNLP, '13))

Build sentence representations using the parse tree to compose embeddings via a nonlinear function taking pairs (c_1, c_2) and output p.





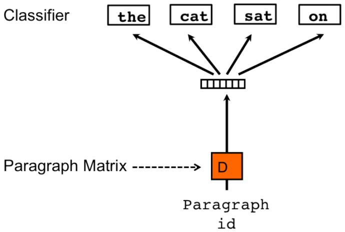
$$s = W^{score} p \quad (9)$$
$$p = f(W[c_1; c_2] + b)$$

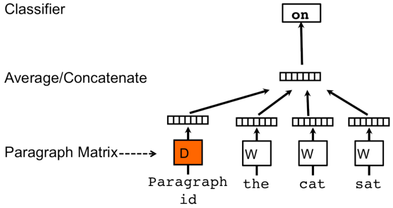
Paragraph Vector

(Le & Mikolov, '14)

A Paragraph Vector (a vector that represents a paragraph/doc) learned by:

1) Predicting the words in a doc; 2) predict *n*-grams in the doc:





At test time, for a new document, one needs to learn its vector, this can encode word order via the *n*-gram prediction approach.

Comparison of CNN, RNN & PV (Kim '14)

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Table 2: Results of our CNN models against other methods. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **Paragraph-Vec**: Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). **CCAE**: Combinatorial Category Autoencoders with combinatorial category grammar operators (Hermann and Blunsom, 2013). **Sent-Parser**: Sentiment analysis-specific parser (Dong et al., 2014). **NBSVM, MNB**: Naive Bayes SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning (2012). **G-Dropout, F-Dropout**: Gaussian Dropout and Fast Dropout from Wang and Manning (2013). **Tree-CRF**: Dependency tree

Some More Recent Work

- Compositionality approaches by Marco Baroni's group:
  Words are combined with linear matrices dependendent on the P.O.S.:
  G. Dinu and M. Baroni. How to make words with vectors: Phrase generation in distributional semantics. ACL '14.

- Document representation by Phil Blunson's group:
  Variants of convolutional networks for text:
  Kalchbrenner et al. A Convolutional Neural Network for Modelling Sentences. ACL '14

Good tutorial slides from these teams covering multiple topics:
New Directions in Vector Space Models of Meaning
http://www.cs.ox.ac.uk/files/6605/aclVectorTutorial.pdf

Summary

- Generic end-to-end deep learning system for NLP tasks.
- Word embeddings combined to form sentence or document embeddings can perform well on supervised tasks.
- Previous common belief in NLP: engineering syntactic features necessary for semantic tasks.
  One can do well by engineering a model/algorithm rather than features.

Attitude is changing in recent years... let's see what happens!

Embedding Methods for NLP

Part 2: Embeddings for Multi-relational Data

Antoine Bordes & Jason Weston
Facebook AI Research

EMNLP tutorial – October 29, 2014

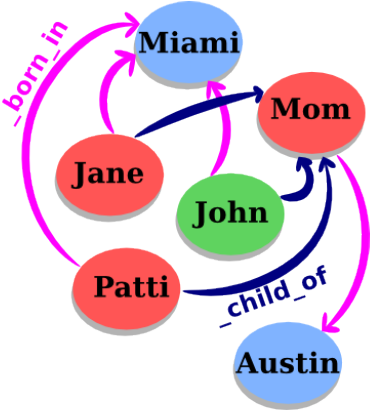
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering
2. Pros and cons of embedding models
3. Future of embedding models
4. Resources

Menu – Part 2

Multi-relational data

- Data is structured as a graph
- Each node = an entity
- Each edge = a relation/fact
- A relation = (sub, rel, obj):
  - sub = subject,
  - rel = relation type,
  - obj = object.
- Nodes w/o features.

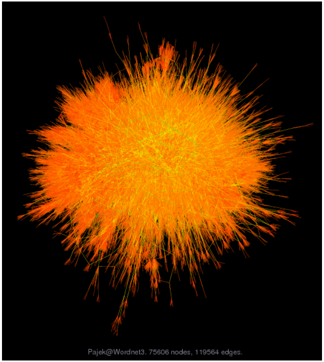


In this talk, we focus on Knowledge Bases (KBs).

Example of KB: WordNet

- WordNet: dictionary where each entity is a sense (synset).

- Popular in NLP.
- Statistics:
  - 117k entities;
  - 20 relation types;
  - 500k facts.
- Examples:
  (car_NN_1, _has_part, _wheel_NN_1)
  (score_NN_1, _is_a, _rating_NN_1)
  (score_NN_2, _is_a, _sheet_music_NN_1)



Example of KB: Freebase

- Freebase: huge collaborative (hence noisy) KB.

- Part of the Google Knowledge Graph.
- Statistics:
  - 80M of entities;
  - 20k relation types;
  - 1.2B facts.

- Examples:
  (Barack Obama, _place_of_birth, Hawai)
  (Albert Einstein, _follows_diet, Veganism)
  (San Francisco, _contains, Telegraph Hill)

Modeling Knowledge Bases

- **Why KBs?**
  - KBs: Semantic search, connect people and things
  - KBs $\leftarrow$ Text: Information extraction
  - KBs $\rightarrow$ Text: Text interpretation, summary, Q&A
- **Main issue:** KBs are hard to manipulate
  - Large dimensions: $10^5/10^8$ entities, $10^4/10^6$ rel. types
  - Sparse: few valid links
  - Noisy/incomplete: missing/wrong relations/entities
- **How?**
  1. Encode KBs into low-dimensional vector spaces
  2. Use these representations:
     - to complete/visualize KBs
     - as KB data in text applications

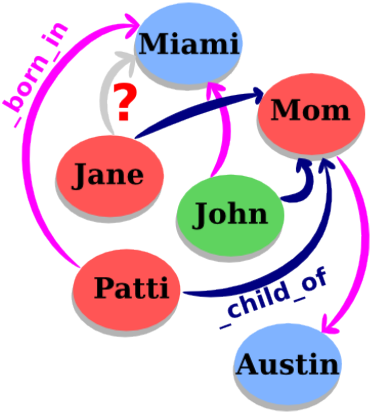
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering
2. Pros and cons of embedding models
3. Future of embedding models
4. Resources

Link Prediction

Add new facts without requiring extra knowledge

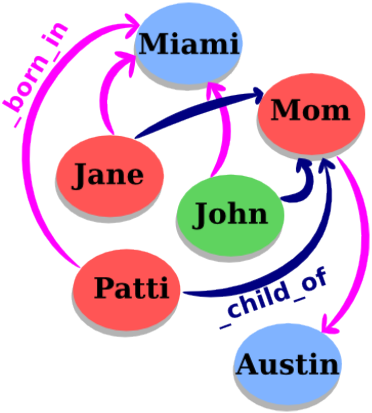
From known information, assess the validity of an unknown fact



Link Prediction

Add new facts without requiring extra knowledge

From known information, assess the validity of an unknown fact



→ *collective classification*
→ *reasoning in embedding spaces*

Statistical Relational Learning

- **Framework:**
  - n_s subjects $\{sub_i\}_{i \in [1;n_s]}$
  - n_r relation types $\{rel_k\}_{k \in [1;n_r]}$
  - n_o objects $\{obj_j\}_{j \in [1;n_o]}$

  $\rightarrow$ For us, $n_s = n_o = n_e$ and $\forall i \in [1; n_e]\,, sub_i = obj_i$.

  - A fact exists for (sub_i, rel_k, obj_j) if $rel_k(sub_i, obj_j) = 1$
- **Goal:** We want to model, from data,

$$\mathbb{P}[rel_k(sub_i, obj_j) = 1]$$

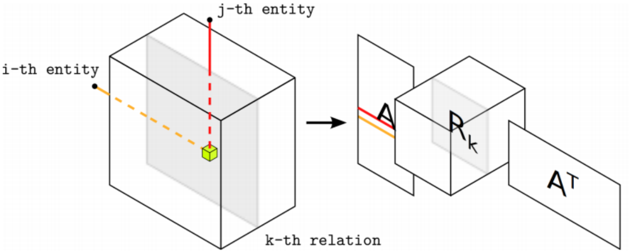
(eq. to approximate the binary tensor $\mathbf{X} \in \{0,1\}^{n_s \times n_o \times n_r}$)

Previous Work

- Tensor factorization (Harshman et al., '94)
- Probabilistic Relational Learning (Friedman et al., '99)
- Relational Markov Networks (Taskar et al., '02)
- Markov-logic Networks (Kok et al., '07)
- Extension of SBMs (Kemp et al., '06) (Sutskever et al., '10)
- Spectral clustering (undirected graphs) (Dong et al., '12)
- Ranking of random walks (Lao et al., '11)
- Collective matrix factorization (Nickel et al., '11)
- Embedding models (Bordes et al., '11, '13) (Jenatton et al., '12) (Socher et al., '13) (Wang et al., '14) (García-Durán et al., '14)

Collective Matrix Factorization (Nickel et al., '11)

- RESCAL: $\forall k \in [1; n_r]$, $\mathbf{R}_k \in \mathbb{R}^{d \times d}$ and $\mathbf{A} \in \mathbb{R}^{n_e \times d}$ (close from DEDICOM (Harshman, '78)).



- **A** & **R** learned by reconstruction (alternating least-squares):

$$\min_{\mathbf{A},\mathbf{R}} \frac{1}{2}\left(\sum_k ||\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^\top||_F^2\right) + \lambda_A||\mathbf{A}||_F^2 + \lambda_R \sum_k ||\mathbf{R}_k||_F^2$$

Scalability

Method	**Nb of parameters**	**on Freebase15k**
RESCAL	$O(n_e d + n_r d^2)$	88M ($d = 250$)

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- RESCAL involves many parameters.
- Bad scalability w.r.t. n_r.
- Reconstruction criterion does not fit well for binary data..

Embedding Models

Two main ideas:

1. Models based on low-dimensional continuous vector embeddings for entities and relation types, directly trained to define a similarity criterion.

2. Stochastic training based on ranking loss with sub-sampling of unknown relations.

Embedding Models for KBs

- Subjects and objects are represented by vectors in $\mathbb{R}^d$.
  - $\{sub_i\}_{i \in [1;n_s]} \quad \rightarrow \quad [\mathbf{s}^1, \ldots, \mathbf{s}^{n_s}] \in \mathbb{R}^{d \times n_s}$
  - $\{obj_j\}_{j \in [1;n_o]} \quad \rightarrow \quad [\mathbf{o}^1, \ldots, \mathbf{o}^{n_o}] \in \mathbb{R}^{d \times n_o}$

  For us, $n_s = n_o = n_e$ and $\forall i \in [1; n_e]\,, \mathbf{s}_i = \mathbf{o}_i$.

- **Rel. types = similarity operators between subj/obj.**
  - $\{rel_k\}_{k \in [1;n_r]} \quad \rightarrow \quad$ operators $\{\mathbf{R}_k\}_{k \in [1;n_r]}$

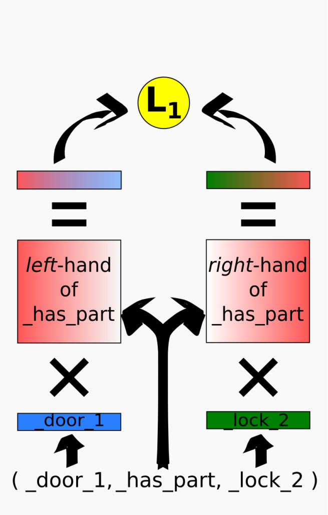
- Learning similarities depending on $rel \rightarrow d(sub, rel, obj)$, parameterized by **s**, **R** and **o**.

Structured Embeddings (Bordes et al., '11)

Intuition: *sub* and *obj* are projected using *rel* in a space where they are similar

$d(sub, rel, obj) = -||\mathbf{R}^{left}\mathbf{s}^\top - \mathbf{R}^{right}\mathbf{o}^\top||_1$

- Entities: **s** and **o** $\in \mathbb{R}^d$
- Projection: $\mathbf{R}^{left}$ and $\mathbf{R}^{right} \in \mathbb{R}^{d\times d}$
  $\mathbf{R}^{left} \neq \mathbf{R}^{right}$ because of asymmetry
- Similarity: L1 distance



Stochastic Training

- Learning by stochastic gradient descent: one training fact after the other
- For each relation from the training set:
  1. sub-sample unobserved facts (false?)
  2. check if the similarity of the true fact is lower
  3. **if not**, update parameters of the considered facts
- Stopping criterion: performance on a validation set

Scalability

Method	Nb of parameters	on Freebase15k
RESCAL	$O(n_e d + n_r d^2)$	88M ($d = 250$)
SE	$O(n_e d + 2 n_r d^2)$	8M ($d = 50$)

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

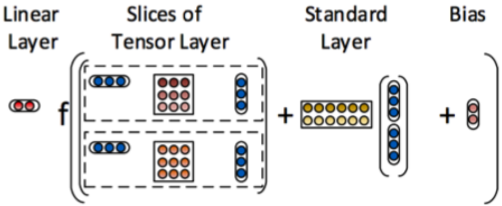
- SE also involves many parameters.
- Bad scalability w.r.t. n_r.
- Potential training problems for SE (overfitting).

Neural Tensor Networks (Socher et al., '13)

- In NTN, a relationship is represented by a tensor, 2 matrices and 2 vectors + a non-linearity (*tanh*).

$$d(sub, rel, obj) = \mathbf{u}_r^\top \tanh\left(\mathbf{h}^\top \mathcal{W}_r \mathbf{t} + \mathbf{V}_r^1 \mathbf{h} + \mathbf{V}_r^2 \mathbf{t} + \mathbf{b}_r\right)$$

- Neural Tensor layer:



- Very powerful model with high capacity for each relation.

Scalability

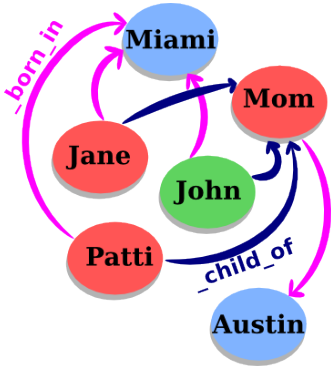
Method	Nb of parameters	on Freebase15k
RESCAL	$O(n_e d + n_r d^2)$	88M ($d = 250$)
SE	$O(n_e d + 2 n_r d^2)$	8M ($d = 50$)
NTN	$O(n_e d + n_r d^3)$	165M ($d = 50$)

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- Very high modeling capacity.
- Involves many parameters.
- Bad scalability w.r.t. n_r (overfitting if few triples).

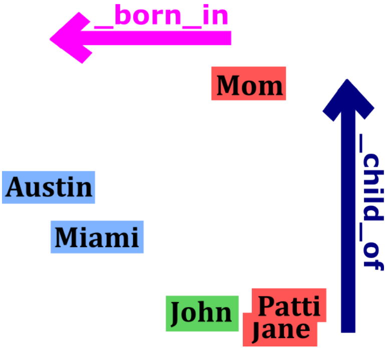
Modeling Relations as Translations (Bordes et al. '13)

Intuition: we want $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$.



Modeling Relations as Translations (NIPS13)

Intuition: we would like that $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$.



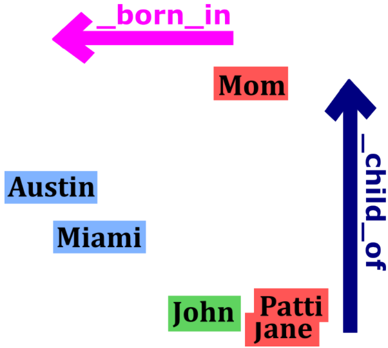
Modeling Relations as Translations (Bordes et al. '13)

Intuition: we want $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$.

The similarity measure is defined as:

$$d(sub, rel, obj) = ||\mathbf{s} + \mathbf{r} - \mathbf{o}||_2^2$$

$\mathbf{s}$,$\mathbf{r}$ and $\mathbf{o}$ are learned to verify that.



Learning TransE

For training, a margin ranking criterion is minimized:

$$\sum_{pos} \sum_{neg \in S'} \left[\gamma + ||\mathbf{s} + \mathbf{r} - \mathbf{o}||_2^2 - ||\mathbf{s'} + \mathbf{r} - \mathbf{o'}||_2^2\right]_+$$

where $[x]_+$ is the positive part of x, $\gamma > 0$ is a margin, and:

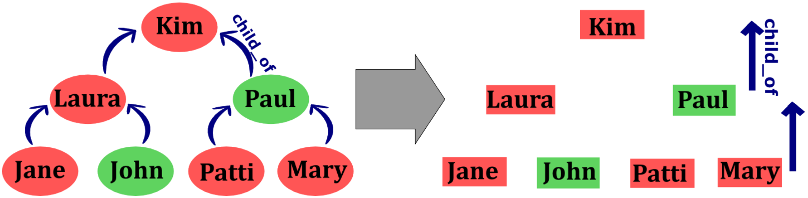
$$S' = \left\{(\text{sub}',\text{rel},\text{obj}) | \text{sub}' \in \mathcal{E}\right\} \cup \left\{(\text{sub},\text{rel},\text{obj}') | \text{obj}' \in \mathcal{E}\right\}$$

Learning TransE

input: Training set $S = \{(\text{sub,rel,obj})\}$, margin γ, learning rate λ
initialize $\mathbf{r} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each rel
$\qquad \mathbf{r} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each ℓ
$\qquad \mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{j}}, \frac{6}{\sqrt{k}})$ for each entity ent(sub or obj)
loop
$\quad \mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity ent
$\quad S_{batch} \leftarrow \textit{sample}(S, b)$ //sample minibatch of size b
$\quad T_{batch} \leftarrow \emptyset$ //initialize set of pairs
$\quad$ **for** $(\text{sub,rel,obj}) \in S_{batch}$ **do**
$\qquad (\text{sub}',\text{rel},\text{obj}') \leftarrow \text{sample}(S'(\text{sub,rel,obj}))$ //sample negative triplet
$\qquad T_{batch} \leftarrow T_{batch} \cup \left\{ \left((\text{sub,rel,obj}), (\text{sub}',\text{rel},\text{obj}') \right) \right\}$
$\quad$ **end for**
$\quad$ Update embeddings w.r.t. $\displaystyle\sum_{T_{batch}} \nabla \left[\gamma + ||\mathbf{s} + \mathbf{r} - \mathbf{o}||_2^2 - ||\mathbf{s'} + \mathbf{r} - \mathbf{o'}||_2^2 \right]_+$
end loop

Motivations of a Translation-based Model

- Natural representation for hierarchical relationships.



- Recent work on word embeddings (Mikolov et al., '13): there may exist embedding spaces in which relationships among concepts are represented by translations.

Scalability

Method	**Nb of parameters**	**on Freebase15k**
RESCAL	$O(n_e d + n_r d^2)$	88M ($d = 250$)
SE	$O(n_e d + 2n_r d^2)$	8M ($d = 50$)
NTN	$O(n_e d + n_r d^3)$	165M ($d = 50$)
TransE	$O(n_e d + n_r d)$	0.8M ($d = 50$)

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- TransE is a special case of SE and NTN.
- TransE obtains better training errors: less overfitting.
- Much better scalability.

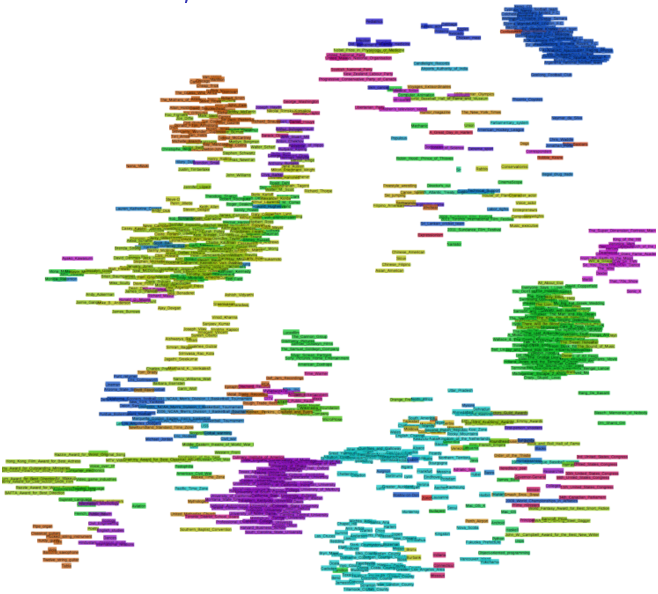
Chunks of Freebase

- **Data statistics**:

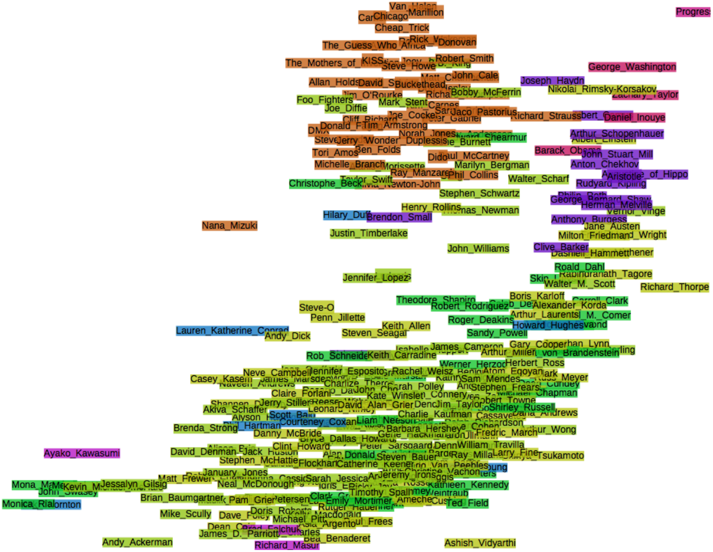
	Entities (n_e)	Rel. (n_r)	Train. Ex.	Valid. Ex.	Test Ex.
FB13	75,043	13	316,232	5,908	23,733
FB15k	14,951	1,345	483,142	50,000	59,071
FB1M	1×10^6	23,382	17.5×10^6	50,000	177,404

- **Training times for** TransE:
  - Embedding dimension: 50.
  - Training time:
    - on Freebase15k: ≈2h (on 1 core),
    - on Freebase1M: ≈1d (on 16 cores).

Visualization of 1,000 Entities

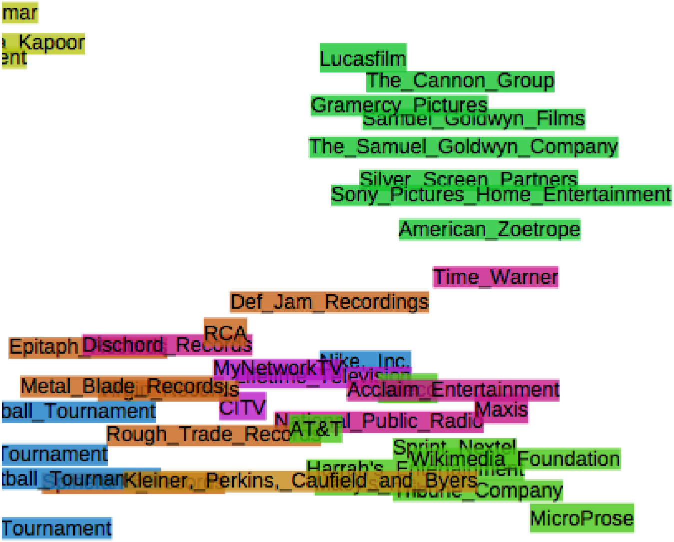


Visualization of 1,000 Entities - Zoom 1



Visualization of 1,000 Entities - Zoom 2

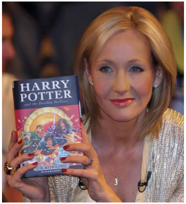
Visualization of 1,000 Entities - Zoom 3



Example

"Who influenced J.K. Rowling?"

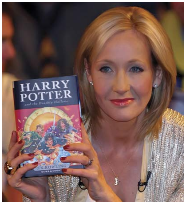
J. K. Rowling `_influenced_by` ?



Example

"Who influenced J.K. Rowling?"

J. K. Rowling _influenced_by G. K. Chesterton
J. R. R. Tolkien
C. S. Lewis
Lloyd Alexander
Terry Pratchett
Roald Dahl
Jorge Luis Borges
Stephen King
Ian Fleming



Example

"Which genre is the movie WALL-E?"

WALL-E _has_genre ?

Example

"Which genre is the movie WALL-E?"

WALL-E _has_genre Animation

Computer animation

Comedy film

Adventure film

Science Fiction

Fantasy

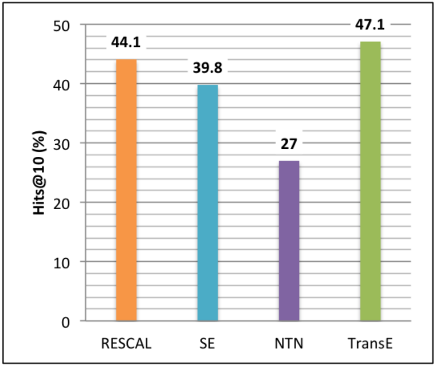
Stop motion

Satire

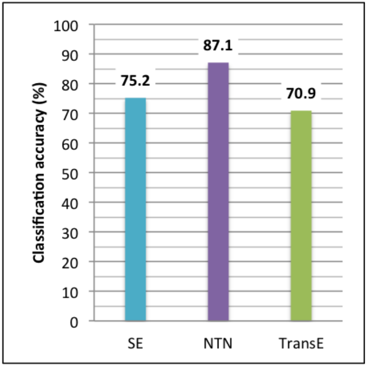
Drama

Benchmarking

Ranking on FB15k



Classification on FB13



On FB1M,TransE predicts 34% in the Top-10 (SE only 17.5%).

Results extracted from (Bordes et al., '13) and (Wang et al., '14)

Refining TransE

- TATEC (García-Durán et al., '14) supplements TransE with a trigram term for encoding complex relationships:

$$d(sub, rel, obj) = \overbrace{\mathbf{s}_1^\top \mathbf{R} \mathbf{o}_1}^{\text{trigram}} + \overbrace{\mathbf{s}_2^\top \mathbf{r} + \mathbf{o}_2^\top \mathbf{r}' + \mathbf{s}_2^\top \mathbf{D} \mathbf{o}_2}^{\text{bigrams}\approx\text{TransE}},$$

with $\mathbf{s}_1 \neq \mathbf{s}_2$ and $\mathbf{o}_1 \neq \mathbf{o}_2$.

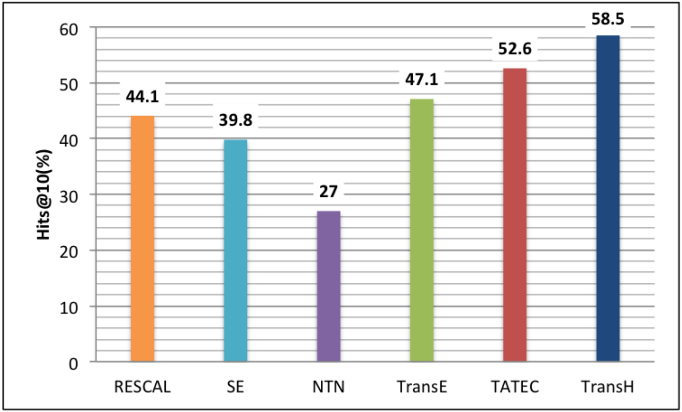
- TransH (Wang et al., '14) adds an orthogonal projection to the translation of TransE:

$$d(sub, rel, obj) = ||(\mathbf{s} - \mathbf{r}_p^\top \mathbf{s} \mathbf{r}_p) + \mathbf{r}_t - (\mathbf{o} - \mathbf{r}_p^\top \mathbf{o} \mathbf{r}_p)||_2^2,$$

with $\mathbf{r}_p \perp \mathbf{r}_t$.

Benchmarking

Ranking on FB15k



Results extracted from (García-Durán et al., '14) and (Wang et al., '14)

Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering
2. Pros and cons of embedding models
3. Future of embedding models
4. Resources

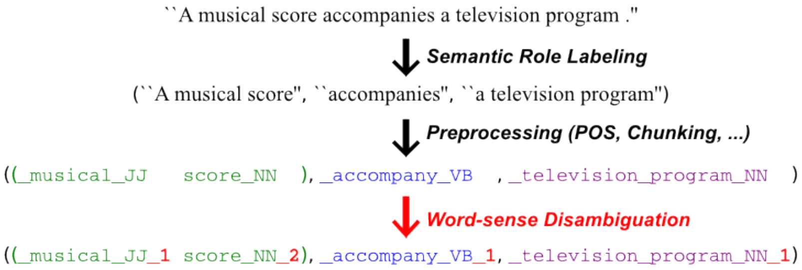
Information Extraction

- Information extraction: populate KBs with new facts using text
- Usually two steps:
  - **Entity linking**: identify mentions of entities in text
  - **Relation extraction**: extract facts about them
- Previous works include rule-based models, classifiers with features from parsers, graphical models, etc.
- Embedding models exist for both steps.

Entity Linking as WSD

Word Sense Disambiguation ↔ WordNet entity linking

Towards open-text semantic parsing:



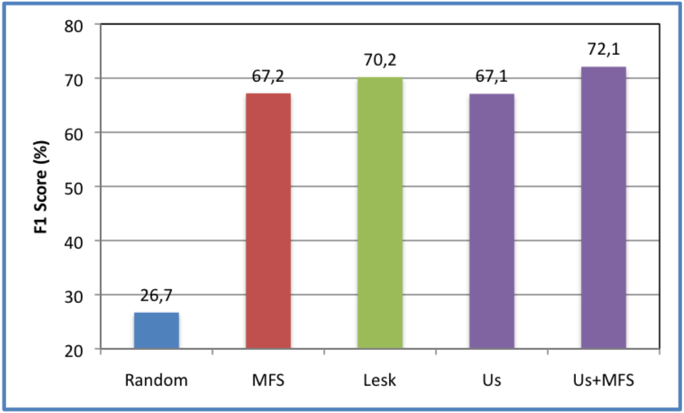
Embeddings of Text and WordNet (Bordes et al., '12)

- Text is converted into relations (*sub*,*rel*,*obj*).
- Joint learning of embeddings for all symbols: words, entities and relation types from WordNet.
- This system can label 37,141 words with 40,943 synsets.

	Train. Ex.	Test Ex.	Labeled?	Symbol
WordNet	146,442	5,000	No	synsets
Wikipedia	2,146,131	10,000	No	words
ConceptNet	11,332	0	Non	words
Ext. WordNet	42,957	5,000	Yes	words+synsets
Unamb. Wikip.	981,841	0	Yes	words+synsets
TOTAL	3,328,703	20,000	-	-

Benchmarking on Extended WordNet

F1-score on 5,000 test sentences to disambiguate.



Results extracted from (Bordes et al., '12)

WordNet is enriched through text

Similarities among senses beyond WordNet

"what does an army attack?"

army_NN_1 `attack_VB_1` ?

WordNet is enriched through text

Similarities among senses beyond original WordNet data

"what does an army attack?"

army_NN_1 `attack_VB_1` troop_NN_4
armed_service_NN_1
ship_NN_1
territory_NN_1
military_unit_NN_1

WordNet is enriched through text

Similarities among senses beyond WordNet

"Who or what earns money"

? `earn_VB_1` money_NN_1

WordNet is enriched through text

Similarities among senses beyond original WordNet data

"Who or what earns money"

person_NN_1	`earn_VB_1`	money_NN_1
business_firm_NN_1		
family_NN_1		
payoff_NN_3		
card_game_NN_1		

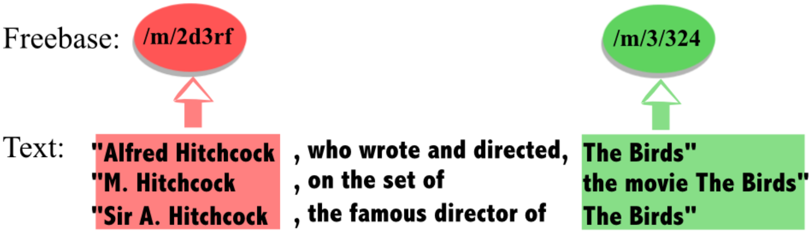
Relation Extraction

Given a bunch of sentences.

Text:			
	"Alfred Hitchcock	**, who wrote and directed,**	**The Birds"**
	"M. Hitchcock	**, on the set of**	**the movie The Birds"**
	"Sir A. Hitchcock	**, the famous director of**	**The Birds"**

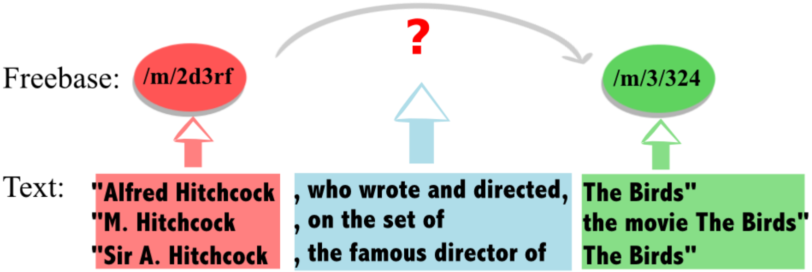
Relation Extraction

Given a bunch of sentences concerning the same pair of entities.



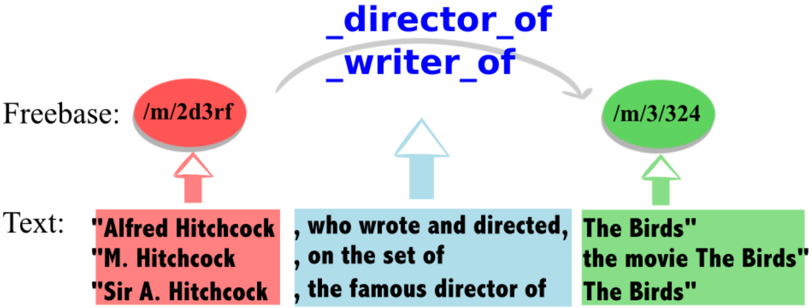
Relation Extraction

Goal: identify if there is a relation between them to add to the KB.



Relation Extraction

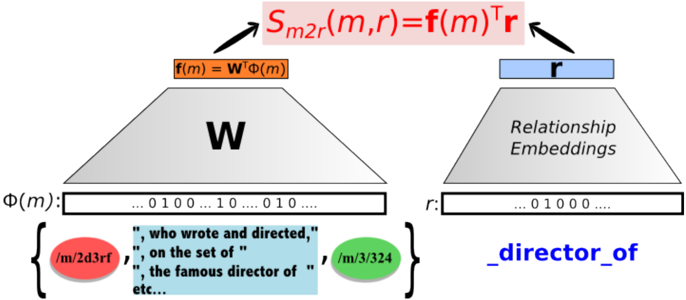
And from which type, to enrich an existing KB.



Embeddings of Text and Freebase (Weston et al., '13)

- **Standard Method:** an embedding-based classifier is trained to predict the relation type, given text mentions $\mathcal{M}$ and (*sub*, *obj*):

$$r(m, sub, obj) = \arg\max_{rel'} \sum_{m \in \mathcal{M}} S_{m2r}(m, rel')$$



Classifier based on WSABIE (Weston et al., '11).

Embeddings of Text and Freebase (Weston et al., '13)

- **Idea:** improve extraction by using both text + available knowledge (= current KB).

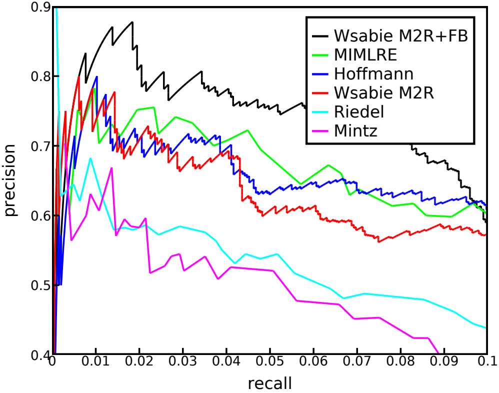
- A model of the KB is used in a re-ranking setting to force extracted relations to agree with it:

$$r'(m, sub, obj) = \arg\max_{rel'} \Big(\sum_{m \in \mathcal{M}} S_{m2r}(m, rel') - d_{KB}(sub, rel', obj) \Big)$$

with $d_{KB}(sub, rel', obj) = ||\mathbf{s} + \mathbf{r}' - \mathbf{o}||_2^2$ (trained separately)

Benchmarking on NYT+Freebase

Exp. on NY Times papers linked with Freebase (Riedel et al., '10)

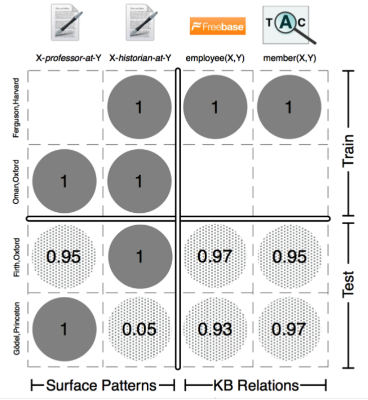


Precision/recall curve for predicting relations

Results extracted from (Weston et al., '13)

Universal Schemas (Riedel et al., '13)

- Join in a single learning problem:
  - relation extraction
  - link prediction
- The same model score triples:
  - made of text mentions
  - from a KB



Universal Schemas (Riedel et al., '13)

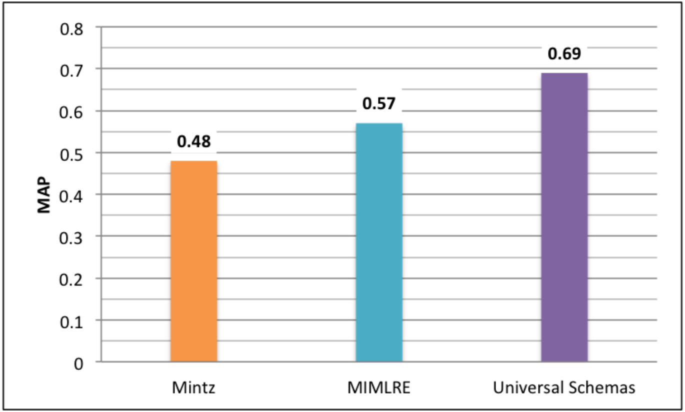
- Relation prediction using the score:

$$r'(m, sub, obj) = \arg\max_{rel'} \Big(\sum_{m \in \mathcal{M}} S_{m2r}(m, rel') + S_{KB}(sub, rel', obj) + S_{neighbors}(sub, rel', obj) \Big)$$

- All scores are defined using embeddings:
  - $S_{m2r}(m, rel') = \mathbf{f}(m)^\top \mathbf{r}'$
  - $S_{kb}(sub, rel', obj) = \mathbf{s}^\top \mathbf{r}'_s + \mathbf{o}^\top \mathbf{r}'_o$
  - $S_{neighbors}(sub, rel', obj) = \sum_{\substack{(sub, rel'', obj) \\ rel'' \neq rel'}} w^{rel'}_{rel''}$
- Training by ranking observed facts versus other and updating using SGD.

Benchmarking on NYT+Freebase

Exp. on NY Times papers linked with Freebase (Riedel et al., '10)



Mean Averaged Precision for predicting relations

Results extracted from (Riedel et al., '13)

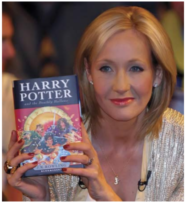
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering
2. Pros and cons of embedding models
3. Future of embedding models
4. Resources

Link Prediction as Q&A

"Who influenced J.K. Rowling?"

J. K. Rowling `_influenced_by` G. K. Chesterton



J. R. R. Tolkien

C. S. Lewis

Lloyd Alexander

Terry Pratchett

Roald Dahl

Jorge Luis Borges

Can we go beyond such rigid structure?

Open-domain Question Answering

- **Open-domain Q&A**: answer question on any topic
  $\longrightarrow$ query a KB with natural language

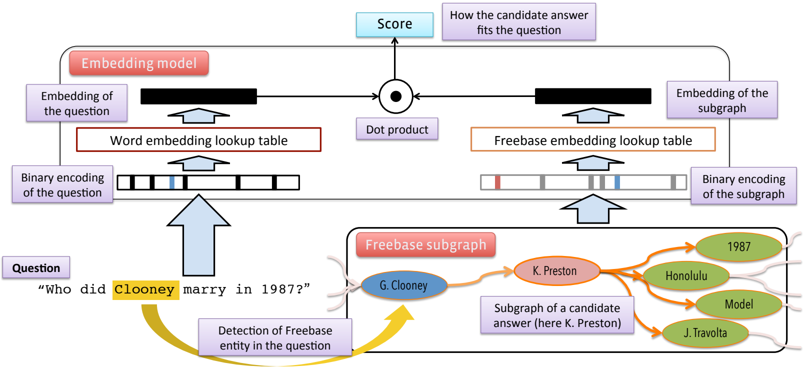
Examples

"What is `cher`'s son's name ?"	`elijah_blue_allman`
"What are `dollars` called in `spain` ?"	`peseta`
"What is `henry_clay` known for ?"	`lawyer`
"Who did `georges_clooney` marry in `1987` ?"	`kelly_preston`

- Recent effort with semantic parsing (Kwiatkowski et al. '13) (Berant et al. '13, '14) (Fader et al., '13, '14) (Reddy et al., '14)
- Models with embeddings as well (Bordes et al., '14)

Subgraph Embeddings (Bordes et al., '14)

- Model learns embeddings of questions and (candidate) answers
- Answers are represented by entity and its neighboring subgraph



Training data

- Freebase is automatically converted into Q&A pairs
- Closer to expected language structure than triples

Examples of Freebase data

(sikkim, location.in_state.judicial_capital, gangtok)
what is the judicial capital of the in state sikkim ? – gangtok

(brighouse, location.location.people_born_here, edward_barber)
who is born in the location brighouse ? – edward_barber

(sepsis, medicine.disease.symptoms, skin_discoloration)
what are the symptoms of the disease sepsis ? – skin_discoloration

Training data

- All Freebase questions have rigid and similar structures
- Supplemented by pairs from clusters of paraphrase questions
- Multitask training: similar questions $\leftrightarrow$ similar embeddings

Examples of paraphrase clusters

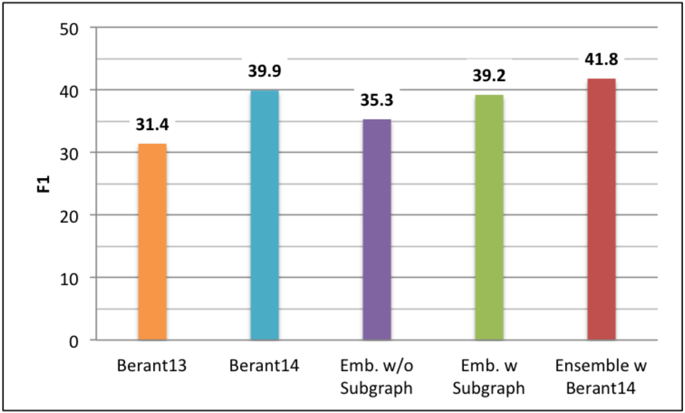
what are two reason to get a 404 ?
what is error 404 ?
how do you correct error 404 ?

what is the term for a teacher of islamic law ?
what is the name of the religious book islam use ?
who is chief of islamic religious authority ?

what country is bueno aire in ?
what countrie is buenos aires in ?
what country is bueno are in ?

Benchmarking on WebQuestions

Experiments on WebQuestions (Berant et al., '13)



F1-score for answering test questions

Results extracted from (Berant et al., '14) and (Bordes et al., '14)

Menu – Part 2

Advantages

- Efficient features for many tasks in practice
- Training with SGD scales & parallelizable (Niu et al., '11)
- Flexible to various tasks: multi-task learning of embeddings
- Supervised or unsupervised training
- Allow to use extra-knowledge in other applications

Issues

- Must train all embeddings together (no parallel 1-vs-rest)
- Low-dimensional vector $\longrightarrow$ compression, blurring
- Sequential models suffer from long-term memory
- Embeddings need quite some updates to be good – not 1-shot
- Negative example sampling can be unefficient

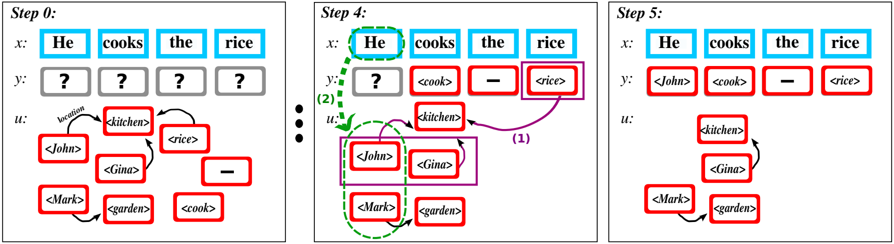
Menu – Part 2

Fix current limitations

- Compression: improve the memory capacity of embeddings and allows for one-shot learning of new symbols
- Long-term memory: encode longer dependencies in sequential models like RNNs
- Training: faster and better sampling of examples
- Beyond linear: most supervised labeling problems are well tackled by simple linear models. Non-linearity should help more.

Explore new directions

- Compositionality (Baroni et al. '10) (Grefenstette, 13)
- Multimodality (Bruni et al., 12) (Kiros et al., '14)
- Grounding language into actions (Bordes et al., 10)



At EMNLP

Modeling Interestingness with Deep Neural Networks
Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He and Li Deng

Translation Modeling with Bidirectional Recurrent Neural Networks
Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker and Hermann Ney

Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics
Douwe Kiela and Léon Bottou

Learning Abstract Concept Embeddings from Multi-Modal Data: Since You Probably Can't See What I Mean
Felix Hill and Anna Korhonen

Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases
Matt Gardner, Partha Talukdar, Jayant Krishnamurthy and Tom Mitchell

Composition of Word Representations Improves Semantic Role Labelling
Michael Roth and Kristian Woodsend

At EMNLP

A Neural Network for Factoid Question Answering over Paragraphs
Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher and Hal Daumé III

Joint Relational Embeddings for Knowledge-based Question Answering
Min-Chul Yang, Nan Duan, Ming Zhou and Hae-Chang Rim

Evaluating Neural Word Representations in Tensor-Based Compositional Settings
Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh and Matthew Purver

Opinion Mining with Deep Recurrent Neural Networks
Ozan Irsoy and Claire Cardie

The Inside-Outside Recursive Neural Network model for Dependency Parsing
Phong Le and Willem Zuidema

A Fast and Accurate Dependency Parser using Neural Networks
Danqi Chen and Christopher Manning

At EMNLP

Reducing Dimensions of Tensors in Type-Driven Distributional Semantics
Tamara Polajnar, Luana Fagarasan and Stephen Clark

Word Semantic Representations using Bayesian Probabilistic Tensor Factorization
Jingwei Zhang, Jeremy Salwen, Michael Glass and Alfio Gliozzo

Glove: Global Vectors for Word Representation
Jeffrey Pennington, Richard Socher and Christopher Manning

Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures
Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa and Yoshimasa Tsuruoka

Typed Tensor Decomposition of Knowledge Bases for Relation Extraction
Kai-Wei Chang, Wen-tau Yih, Bishan Yang and Christopher Meek

Knowledge Graph and Text Jointly Embedding
Zhen Wang, Jianwen Zhang, Jianlin Feng and Zheng Chen

At EMNLP

Question Answering with Subgraph Embeddings
Antoine Bordes, Sumit Chopra and Jason Weston

Word Translation Prediction for Morphologically Rich Languages with Bilingual Neural Networks
Ke M. Tran, Arianna Bisazza and Christof Monz

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio

Convolutional Neural Networks for Sentence Classification
Yoon Kim

#TagSpace: Semantic Embeddings from Hashtags
Jason Weston, Sumit Chopra and Keith Adams

Menu – Part 2

Code

- Torch: `www.torch.ch`
- SENNA: `ronan.collobert.com/senna`
- RNNLM: `www.fit.vutbr.cz/~imikolov/rnnlm`
- Word2vec: `code.google.com/p/word2vec`
- Recursive NN: `nlp.stanford.edu/sentiment`
- SME (multi-relational data): `github.com/glorotxa/sme`

References

Semantic Parsing on Freebase from Question-Answer Pairs
J. Berant, A. Chou, R. Frostig & P. Liang. *EMNLP, 2013*

Semantic Parsing via Paraphrasing
J. Berant & P. Liang. *ACL, 2013*

Learning Structured Embeddings of Knowledge Bases
A. Bordes, J. Weston, R. Collobert & Y. Bengio. *AAAI, 2011*

Joint Learning of Words and Meaning Rep. for Open-Text Semantic Parsing
A. Bordes, X. Glorot, J. Weston & Y. Bengio. *AISTATS, 2012*

A Semantic Matching Energy Function for Learning with Multi-relational Data
A. Bordes, X. Glorot, J. Weston & Y. Bengio. *MLJ, 2013*

Translating Embeddings for Modeling Multi-relational Data
A. Bordes, N. Usunier, A. García-Durán, J. Weston & O. Yakhnenko. *NIPS, 2013*

References

Question Answering with Subgraph Embeddings
A. Bordes, S. Chopra & J. Weston. *EMNLP, 2014*

Clustering with Multi-Layer Graphs: A Spectral Perspective
X. Dong, P. Frossard, P. Vandergheynst & N. Nefedov. *IEEE TSP, 2013*

Paraphrase-Driven Learning for Open Question Answering
A. Fader, L. Zettlemoyer & O. Etzioni. *ACL, 2013*

Open Question Answering Over Curated and Extracted Knowledge Bases
A. Fader, L. Zettlemoyer & O. Etzioni. *KDD, 2014*

Learning Probabilistic Relational Models
N. Friedman, L. Getoor, D. Koller & A. Pfeffer. *IJCAI, 1999*

Effective Blending of Two and Three-way Interactions for Modeling Multi-relational Data
A. García-Durán, A. Bordes & N. Usunier. *ECML-PKDD, 2014*

References

Models for the Analysis of Asymmetrical Relationships among n Objects or Stimuli
R. Harshman. *Joint Symposium of Psych. and Mathematical Societies, 1978*.

PARAFAC: Parallel factor analysis
R. Harshman & M. Lundy. *Comp. Statistics and Data Analysis, 1994*

A Latent Factor Model for Highly Multi-relational Data
R. Jenatton, N. Le Roux, A. Bordes & G. Obozinski. *NIPS, 2012*.

Learning Systems of Concepts with an Infinite Relational Model
C. Kemp, J. Tenenbaum, T. Griffiths, T. Yamada & N. Ueda. *AAAI, 2006.*

Statistical Predicate Invention
S. Kok, P. Domingos. *ICML, 2007*

Scaling Semantic Parsers with On-the-fly Ontology Matching
T. Kwiatkowski, E. Choi, Y. Artzi & L. Zettlemoyer. *EMNLP, 2013*

References

Random Walk Inference and Learning in A Large Scale Knowledge Base
N. Lao, T. Mitchell & W. Cohen. *EMNLP, 2011*.

Distributed Representations of Words and Phrases and their Compositionality
T. Mikolov, I. Sutskever, K. Chen, G. Corrado & J. Dean. *NIPS, 2013*.

A Three-Way Model for Collective Learning on Multi-Relational Data
M. Nickel, V. Tresp & H.-P. Kriegel. *ICML, 2011*.

Large-scale Semantic Parsing without Question-Answer Pairs
S. Reddy, M. Lapata & M. Steedman. *TACL, 2014.*

Modeling Relations and Their Mentions without Labeled Text
S. Riedel, L. Yao and A. McCallum. *ECML-PKDD, 2010*

Relation Extraction with Matrix Factorization and Universal Schemas
S. Riedel, L. Yao, B. Marlin and A. McCallum. *HLT-NAACl, 2013*

References

Reasoning With Neural Tensor Networks for Knowledge Base Completion
R. Socher, D. Chen, C. Manning & A. Ng *NIPS, 2013*.

Modelling Relational Data using Bayesian Clustered Tensor Factorization
I. Sutskever, R. Salakhutdinov & J. Tenenbaum. *NIPS, 2009*.

Discriminative Probabilistic Models for Relational Data
B. Taskar, P. Abbeel & D. Koller. *UAI, 2002*.

Knowledge Graph Embedding by Translating on Hyperplanes
Z. Wang, J. Zhang, J. Feng & Z. Chen. *AAAI, 2014.*

Wsabie: Scaling Up To Large Vocabulary Image Annotation
J. Weston, S. Bengio & N. Usunier. *IJCAI, 2011*

Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction.
J. Weston, A. Bordes, O. Yakhnenko & N. Usunier. *EMNLP, 2013*