Deep learning for natural language processing

Word representations

Benoit Favre <benoit.favre@univ-amu.fr>

Aix-Marseille Université, LIF/CNRS

03-2018

Motivation

- How to represent words as input of neural network?
  - 1-of-n (or 1-hot)
    - ⋆ Each word form is a dimension in a very large vector (one neuron per possible word)
    - ⋆ It is set to 1 if the word is seen, 0 otherwise
    - ⋆ Typically dimension of 100k
  - A text can then be represented as a matrix of size $(length \times |vocab|)$
- Problems
  - Size is very inefficient (realist web vocab is 1M+)
  - Orthogonal (synonyms have different representations)
  - How to account for unknown words (difficult to generalize on small datasets)

Representation learning

- Motivation for machine-learning based NLP
  - Typically large input space (parser = 500 million dimensions)
  - Low rank: only a smaller number of features useful
  - How to generalize lexical relations?
  - One representation for every task
- Approaches
  - Feature selection (Greedy, information gain)
  - Dimensionality reduction (PCA, SVD, matrix factorization...)
  - Hidden layers of a neural network, autoencoders
- Successful applications
  - Image search (Weston, Bengio et al, 2010)
  - Face identification at Facebook (Taigman et al, 2014)
  - Image caption generation (Vinyals et al, 2014)
  - Speaker segmentation (Rouvier et al, 2015)
  - $\rightarrow$ Word embeddings

Word embeddings

- Objective
  - From one-of-n (or one-hot) representation to low dimensional vectors
  - Similar words should be similarly placed
  - Train from large quantities of text (billions of words)
- Distributional semantic hypothesis
  - Word meaning is defined by their company
  - Two words occurring in the same context are likely to have similar meaning
- Approaches
  - LSA (Deerwester et al, 1990)
  - Random indexing (Kanerva et al, 2000)
  - Corrupted n-gram (Colobert et al, 2008)
  - Hidden state from RNNLM or NNLM (Bengio et al)
  - Word2vec (Mikovol et al, 2013)
  - GloVe (Pennington et al, 2014)

	w_1	w_2	w_3	w_4	w_5
w_1		1		3	
w_2					
w_3	1	2		1	
w_4	2				2
w_5	3	1			

Historical approaches: LSA

- Latent semantic analysis (LSA, 1998)
  - Create a word by document matrix M: $m_{i,j}$ is the log of the frequency of word i in document j.
  - Perform a SVD on the coocurrence matrix $M = U\Sigma V^T$
  - Use U as the new representation (U_i is the representation for word i)
  - Since M is very large, optimize SVD (Lanczos' algorithm...)
  - Extension: build a word-by-word cooccurrence matrix within a moving window

Historical approaches: Random indexing

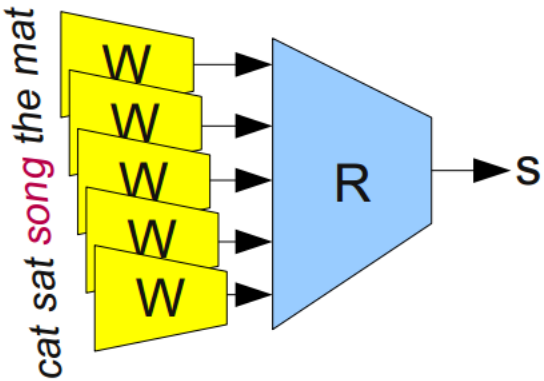
- Random indexing (SahIgren, 2005)
  - Associate each word with a random $n - hot$ vector of dimension m (example: 4 non-null components in a 300-dim vector)
  - It is unlikely that two words have the same representation, so the vectors have a high probability of being an orthogonal basis
  - Create a $|vocab| \times m$ cooccurrence matrix
  - When words i and j cooccur, add the representation for word j to row i
  - This approximates a low-rank version of the real coocurrence matrix
  - After normalization (and optionally PCA), row i can be used as new representation for word i
- Need to scale to very large datasets (billions of words)

Corrupted n-grams

- Approach: learn to discriminate between existing word n-grams and non-existing ones
  - Input: 1-hot representation for each word of the n-gram
  - Output: binary task, whether the n-gram exists or not
  - Parameters W and R (W is shared between word positions)
  - Mix existing n-grams with corrupted n-grams in training data

$$r_i = W x_i \quad \forall i \in [1 \dots n]$$

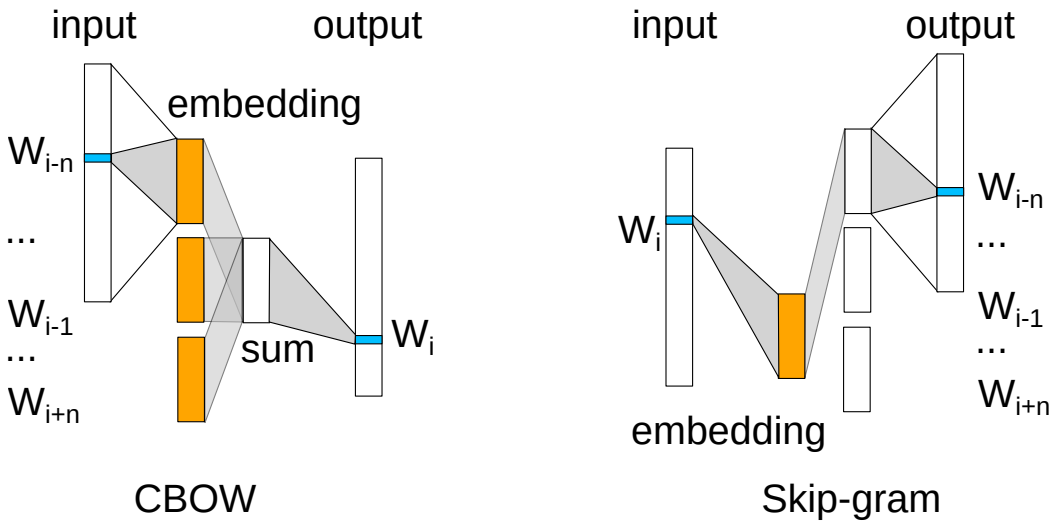
$$y = softmax(R \sum_{i=1}^{n} r_i)$$



- Extension: train any kind of language model
  - Continuous-space language model (CSLM, Schwenk et al)
  - Recurrent language models
  - Multi-task systems (tagging, named entity, chunking, etc)

Word2vec

- Proposed by [Mikolov et al, 2013], code available at `https://github.com/dav/word2vec`.
- Task
  1. Given bag-of-word from window, predict central word (CBOW)
  2. Given central word, predict another word from the window (Skip-gram)



- Training (simplified)
  - For each word-context (x, y) :
    - $\hat{y} = softmax(Wx + b)$
    - Update W and b via error back-propagation

Global vectors (GloVe)

- Main idea (Pennington et al, 2014)
  - $P(k|i)/P(k|j)$ is high when i and j are similar words
  - $\rightarrow$ Find fixed size representations that respect this constraint

$k =$	solid	gas	water	fashion
$P(k\|\text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|\text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|\text{ice})/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

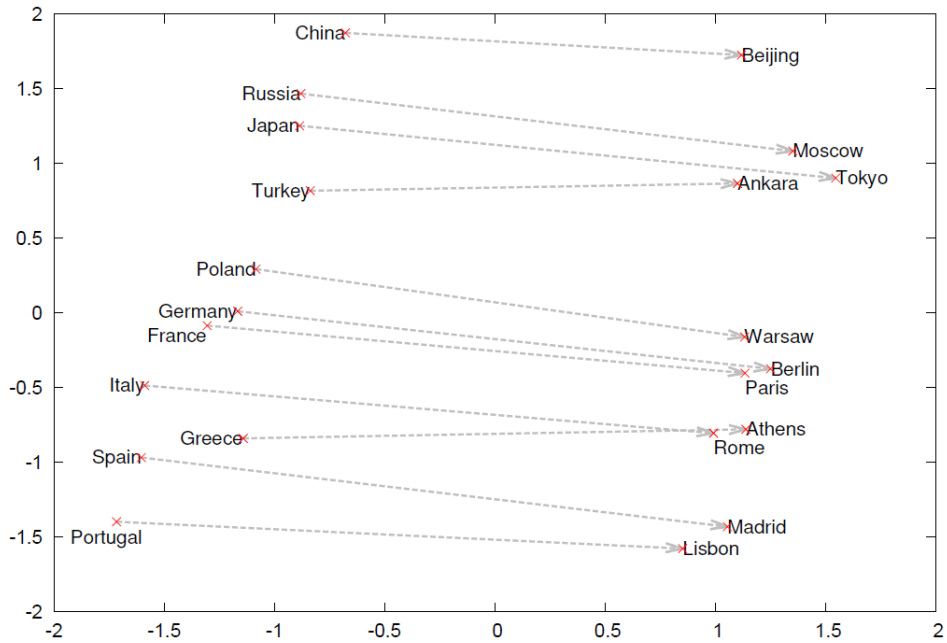
- Training
  - Start from (sparse) cooccurrence matrix $\{m_{ij}\}$
  - Then minimize following loss function

$$Loss = \sum_{i,j} f(m_{ij}) \left(w_i^T w_j + b_i + b_j - \log m_{ij} \right)^2$$

- f dampers the effect of low frequency pairs, in particular $f(0) = 0$
- Worst-case complexity in $|vocab|^2$, but
  - Since $f(0) = 0$ only need to compute for seen coocurrences
  - Linear in corpus size on well-behaved corpora

Linguistic regularities

- Inflection
  - Plural, gender
  - Comparatives, superlatives
  - Verb tense
- Semantic relations
  - Capital / country
  - Leader / group
  - analogies
- Linear relations
  - king + (woman - man) = queen
  - paris + (italy - france) = rome

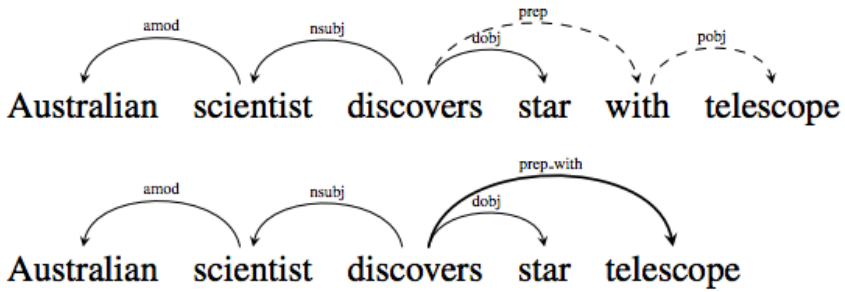


Example[1] trained on comments from www.slashdot.org.

[1] http://pageperso.lif.univ-mrs.fr/~benoit.favre/tsne-slashdot/

Word embedding extensions

- Dependency embeddings (Levy et al, 2014)
  - Use dependency tree instead of context window
  - Represent word with dependents and governor
  - Makes much more syntactic embeddings



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Task-specific embeddings

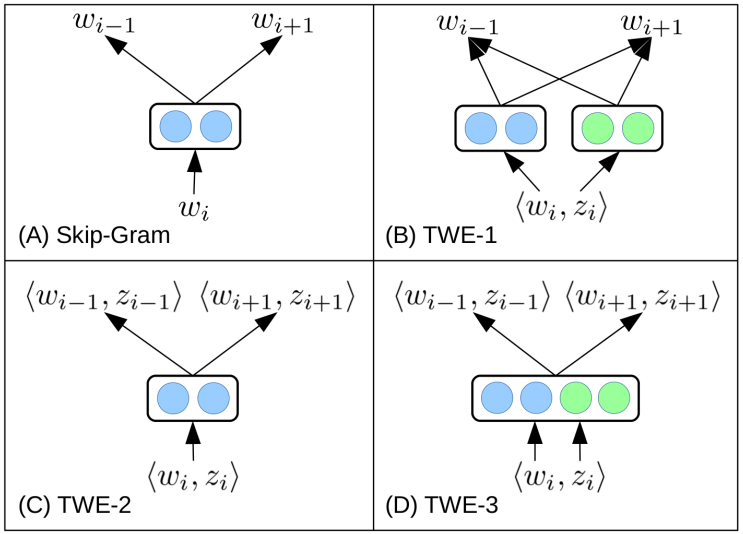
- Variants in embedding training
  - Lexical: words
  - Part-of-speech: joint model for (word, pos-tag)
  - Sentiment: also predict smiley in tweet

Lexical		**Part-of-speech**		**Sentiment**	
good	**bad**	**good**	**bad**	**good**	**bad**
great	good	great	good	great	terrible
bad	terrible	bad	terrible	goid	horrible
goid	baaad	nice	horrible	nice	shitty
gpod	horrible	gd	shitty	goood	crappy
gud	lousy	goid	crappy	gpod	sucky
decent	shitty	decent	baaaad	gd	lousy
agood	crappy	goos	lousy	fantastic	horrid
goood	sucky	grest	sucky	wonderful	stupid
terrible	horible	guid	fickle-minded	gud	:/
gr8	horrid	goo	baaaaad	bad	sucks

- $\rightarrow$ State-of-the art sentiment analysis at SemEval 2016

Sense-aware embeddings

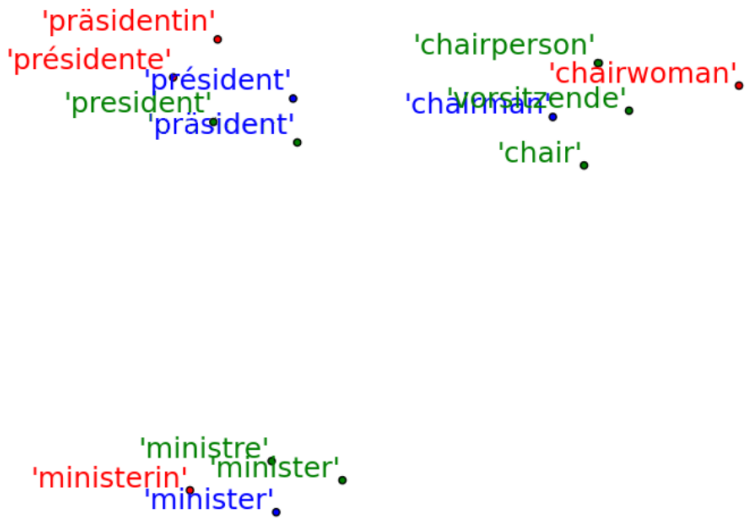
- Multi-prototype embeddings (Huang et al, 2012; Liu et al, 2015)
  - Each word shall have one embedding for each of its senses
  - Hidden variables: a word has n embeddings
  - Can pre-process with topic tagging (LDA)



Word	Prior Probability	Most Similar Words
apple_1	0.82	strawberry, cherry, blueberry
apple_2	0.17	iphone, macintosh, microsoft
bank_1	0.15	river, canal, waterway
bank_2	0.6	citibank, jpmorgan, bancorp
bank_3	0.25	stock, exchange, banking
cell_1	0.09	phones, cellphones, mobile
cell_2	0.81	protein, tissues, lysis
cell_3	0.01	locked, escape, handcuffed

Multilingual embeddings

- Can we create a single embedding space for multiple languages?
  - Train bag-of-word autoencoder on bitexts (Hermann et al, 2014)
    - ★ Force sentence-level representations (bag-of-words) to be similar
    - ★ For instance, sentence representations can be bag-of-words



Mapping embedding spaces

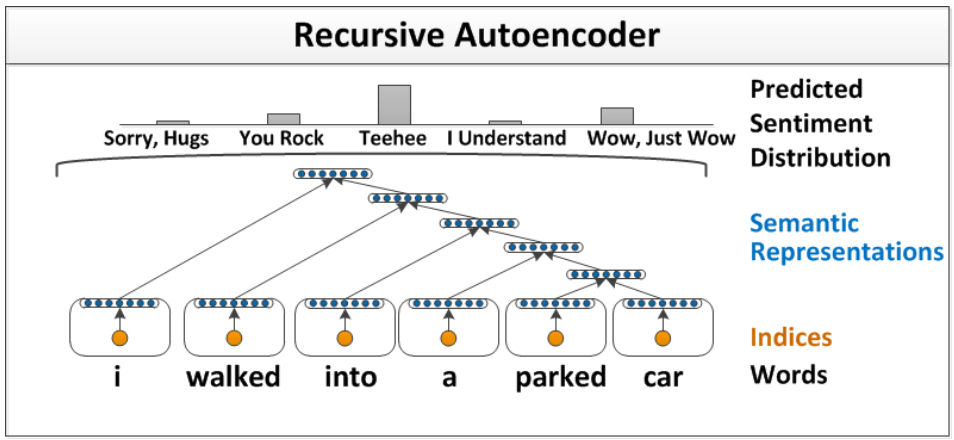
- Problem
  - Infinite number of solutions to "embedding training"
  - Need to map words so that they are in the same location
- Approach
  1. Select common subset of words between two spaces
  2. Find linear transform between them
  3. Apply to remaining words
- Hypotheses
  - Most words do not change meaning
  - Linear transform conserves (linear) linguistic regularities
- Formulation
  - V and W are vector spaces of same dimension, over the same words
  - $V = P \cdot W$ where P is the linear transform matrix
  - Find $P = V \cdot W^{-1}$ using pseudo-inverse
  - Compute mapped representation for all words $W' = P \cdot W_{all}$

Application to cross-lingual NLP

- Use small bilingual dictionary to constrain mapping
  - Space is the same in both languages
- Cross lingual topic modeling
  - Train a classifier to detect topics in source language
  - Map embeddings with bilingual constraint
  - Leads to almost the same performance as a model trained on the target language
- Cross lingual sentiment analysis
  - Can be used to translate sentiment lexicons
- Other applications
  - Track embedding change in time, or across topic

Compositional meaning

- Task-adapted embeddings (Socher et al.)
  - Combine word-level embeddings
  - Follow parse tree, learn constituent-specific combiners
  - Sentence representation is supervised by task (Sentiment analysis)

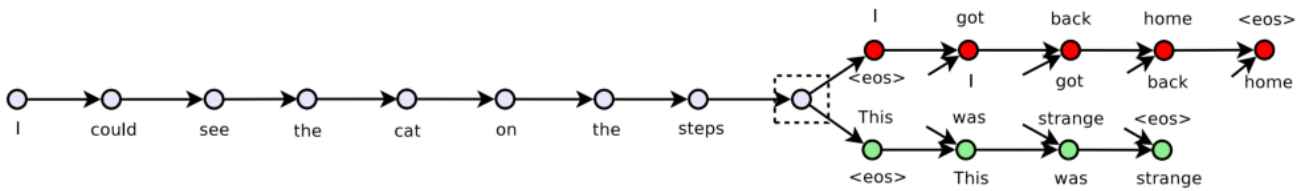


Morphological embeddings

- Account for morphology
  - Generate representations for unseen words
  - Account for the functional role of morphology
- RNN:
  - Accumulate character-level representations
- Predict morphological features [Cotterell et al, 2015]
  - Features: tense, genre, case, animacy...
  - Requires large training set
- Fasttext [Bojanowski et al, 2016]
  - word2vec on (form + character n-grams)
  - (forest, ^fo, for, ore, res, est, st$) $\rightarrow$ context

Sentence and document embeddings

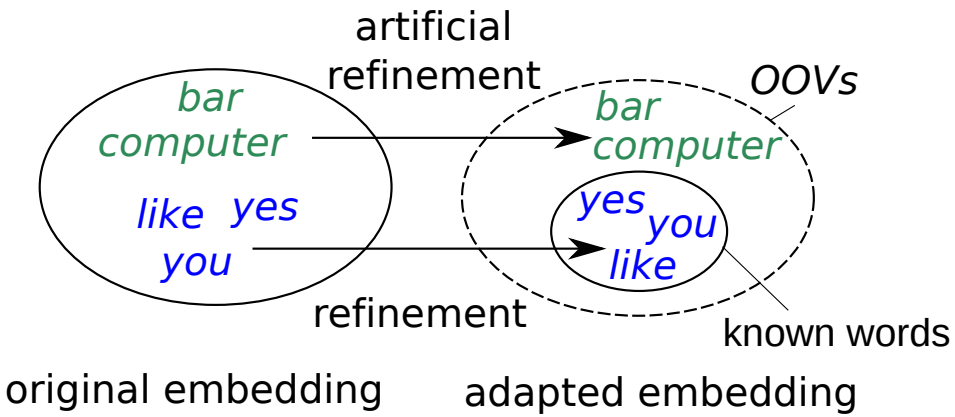
- Skip-Though vectors
  - Train a system to generate the next and previous sentence from the current sentence
  - Sentences that appear in the same context will have similar embeddings



- Doc2vec / paragraph vectors
  - Represent sentences in one-hot vector (very high dimensional)
  - Train word2vec or similar algorithm

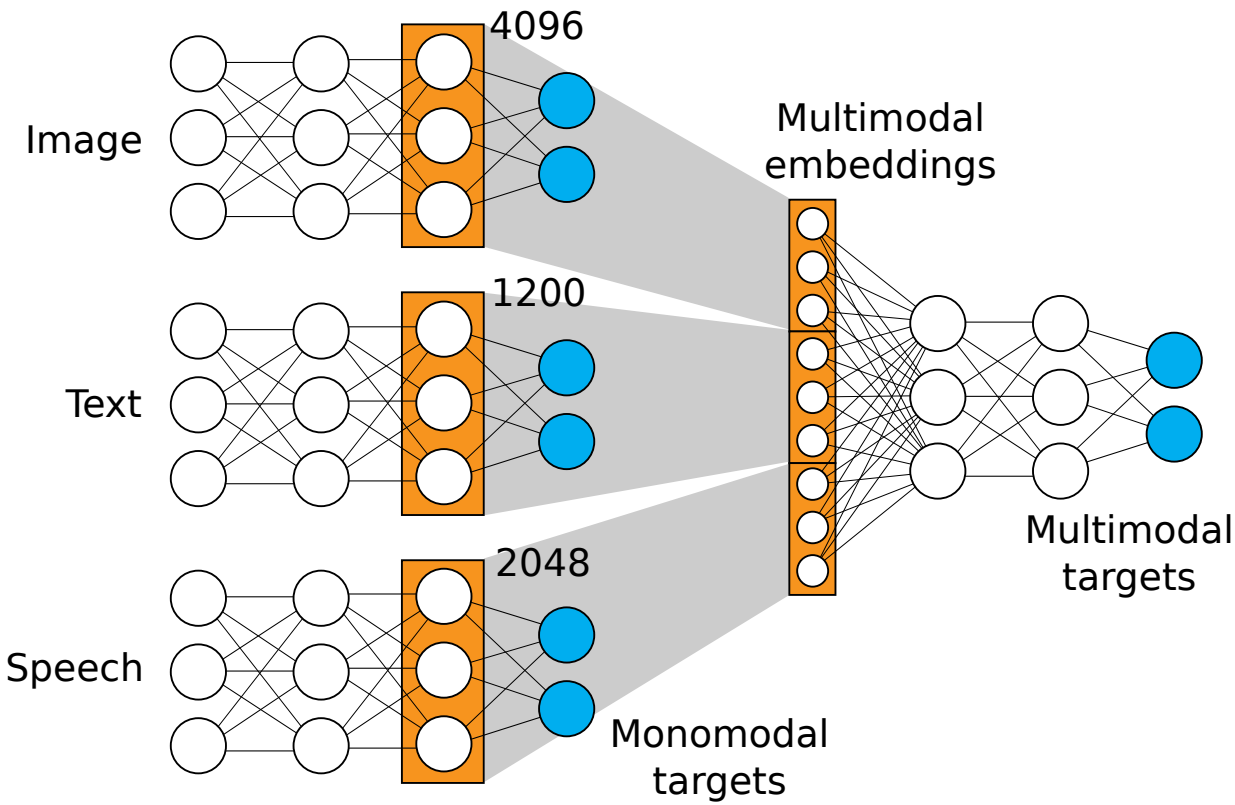
Out-of-vocabulary word handling

- We will use embeddings as representation for training a NLP system
  - Embeddings are refined to the task at hand
- At test time, what can we do with words that we have never seen?
  - OOV_1: They are neither seen when training an NLP system, nor have an embedding
    - ★ Do we have corpus where they occur?
    - ★ Use embedding of closest word in term of edit distance
    - ★ Character embeddings
  - OOV_2: They don't have an embedding but appear in training data
    - ★ Similar to OOV_1
  - OOV_3: They are not in the NLP system training data, but have an embedding
    - ★ Artificially refine the representation



Multimodal embeddings

- Different inputs contribute to a task
  - Speech
  - Image
  - Text
- Pretrain each modality, then generate multimodal embeddings

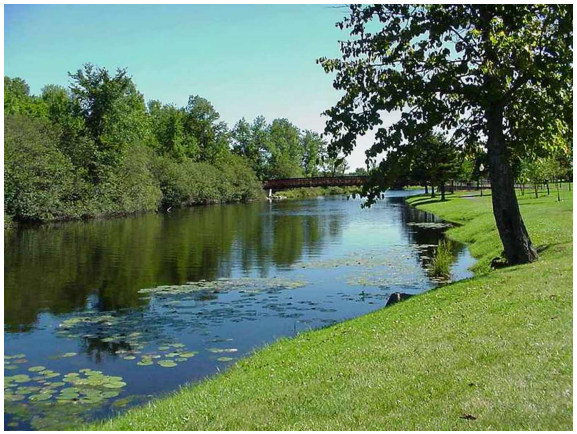


Evaluating embeddings

- What makes good word embeddings?
  - We want embeddings that are general enough to be reused
  - They encode known linguistic properties
  - They encode "relatedness" and "similarity"
  - They lead to good performance when used in a system
- Linguistic properties
  - Compare to Wordnet or Babelnet (`http://babelnet.org/`)
  - Analogies
- Psychological properties
  - Ask human judges to rate the similarity between a pair of words
  - Likert scale 1 to 10
  - 15-30 raters
  - Compute the correlation between cosine similarity and human ratings
- Can we do better?

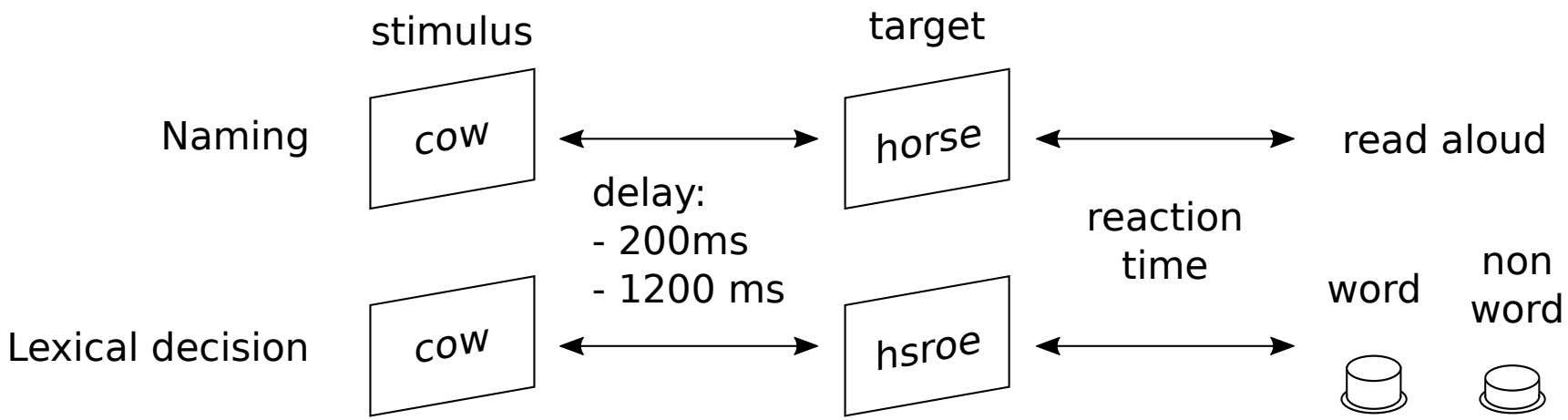
Physiologically plausible embeddings?

- Can we do better?
  - Look at how the brain reacts to stimuli
- Priming effect between two related words
  - Seminal work by Meyer & Schvaneveldt in 1971
  - Decrease of reaction time in a lexical decision task
    - ★ Measure the time needed to decide if a word exists or not after seeing a stimulus
- Can be used to evaluate word embeddings



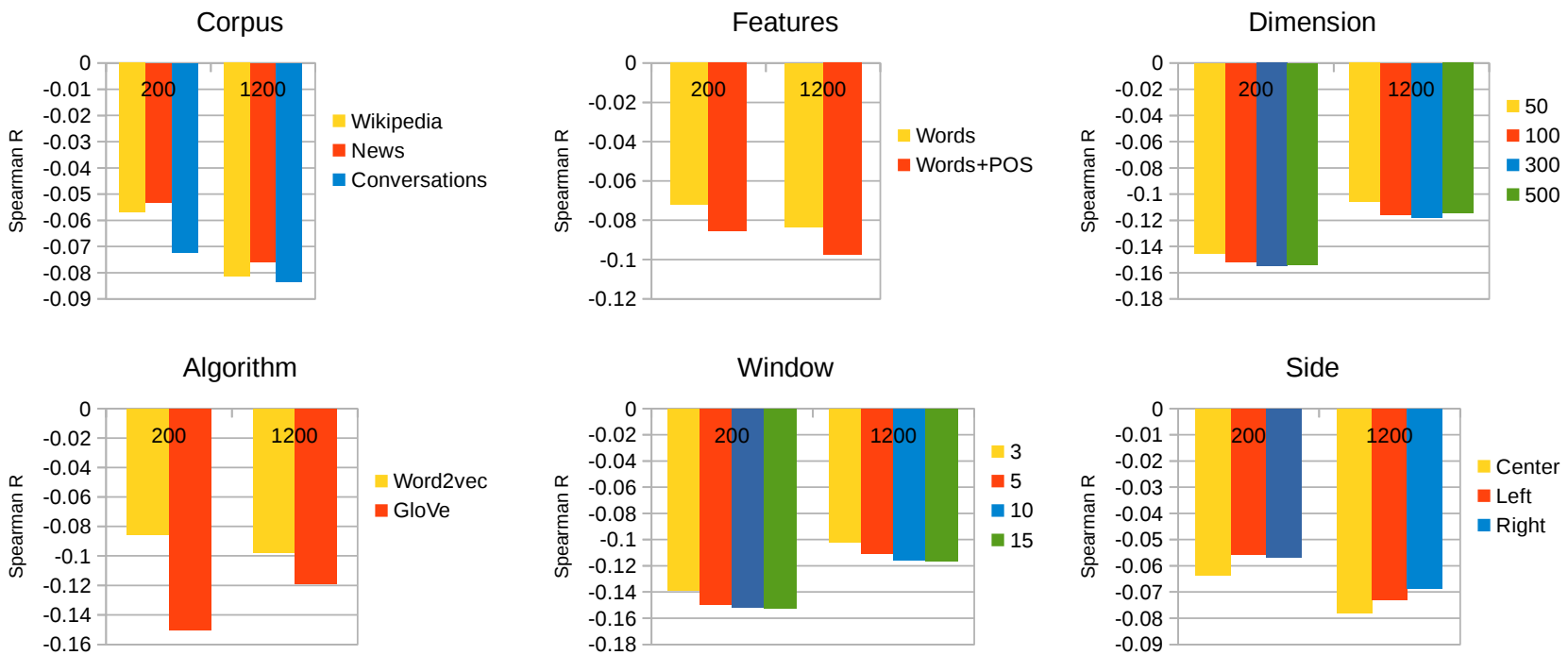
Semantic priming project

- Montana state university (Hutchison et al., 2013)
  - 768 human subjects
  - 1.7 million measures
  - 9 demographics + 3 tests (reading, comprehension...)
  - 6,000 pair of words
  - `http://spp.montana.edu/`
- Experimental protocol



Effect of parameters when learning embeddings

- Negative correlation indicates
  - Shorter reaction times lead to higher cosine similarity



Conclusions

- Representations for words
  - 1-hot is too large, and does not convey relationships between words
  - $\rightarrow$ low-dimensional dense vector
- Methods
  - Word2vec: predict surrounding words given window center
  - GloVe: build an approximation of the cooccurrence matrix
- Extensions
  - Cross-lingual representations
  - Task-specific embeddings
- Evaluation
  - Are word embeddings representative of brain inner working
  - What is the best representation for a given task