Neural Models for Phrase and Sentence Meaning

Dimitri Kartsaklis

School of Electronic Engineering
and Computer Science

Queen Mary
University of London

NLP Course
27 March 2017

In a nutshell

- The problem of producing robust representations for the meaning of phrases and sentences is at the core of every NLP task.
- In this talk, we see how neural architectures can be used towards this goal.
- We provide a generic introduction to NNs, and we review specific architectures such as recursive, recurrent and convolutional neural networks in the context of language.

Outline

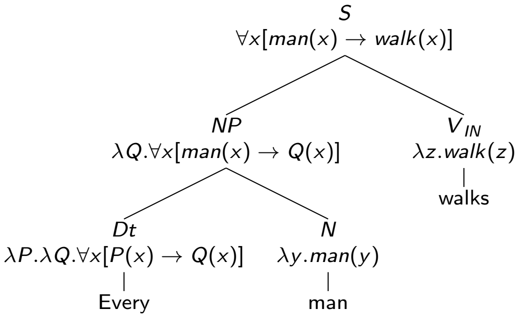
Sentence meaning representation in NLP

Representation of sentence meaning is central to many important NLP tasks:

- **Paraphrase detection:** Given two sentences, decide if they say the same thing in different words
- **Sentiment analysis:** Extract the general sentiment from a sentence or a document
- **Textual entailment:** Decide if one sentence logically infers a different one
- **Machine translation:** Automatically translate one sentence into a different language
- **Document summarization:** Create a summary of a document by extracting its most representative sentences

Formal semantics approach

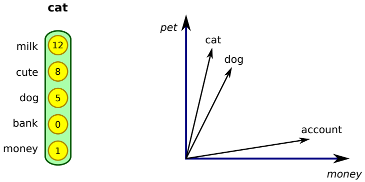
- Logical forms of compounds are computed via β-reduction:



- The semantic value of a sentence is **true** or **false**, and the actual meaning of words remains unspecified.
- Can we do better than that?

From truth-theoretic models to quantitative models

- Let's start at word level: *The meaning of a word is determined by the contexts in which that word occurs* (Harris, 1968)
- So the meaning of a word can be represented as a vector of co-occurrence statistics with a pre-defined set of contexts:

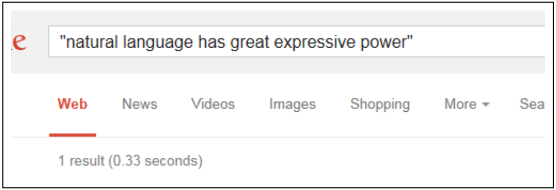


- Semantic relatedness is usually based on cosine similarity:

$$\text{sim}(\overrightarrow{v}, \overrightarrow{u}) = \cos\theta_{\overrightarrow{v},\overrightarrow{u}} = \frac{\langle \overrightarrow{v} \cdot \overrightarrow{u} \rangle}{\|\overrightarrow{v}\|\|\overrightarrow{u}\|}$$

Compositional distributional models

- Distributional models of meaning are quantitative, but they do not scale up to phrases and sentences; there is not enough data:



- However, we can produce a sentence vector by composing the distributional vectors of the words in that sentence. A compositional distributional model is a function of the following form:

$$\overrightarrow{s} = f(\overrightarrow{w_1}, \overrightarrow{w_2}, \ldots, \overrightarrow{w_n})$$

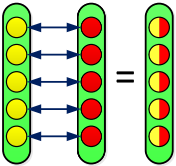
Element-wise composition

- The easiest way to compose two vectors is by working element-wise [Mitchell and Lapata (2010)]:

$$\overrightarrow{w_1 w_2} = \alpha \overrightarrow{w_1} + \beta \overrightarrow{w_2} = \sum_i (\alpha c_i^{w_1} + \beta c_i^{w_2}) \overrightarrow{n_i}$$

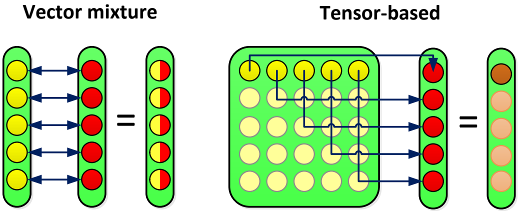
$$\overrightarrow{w_1 w_2} = \overrightarrow{w_1} \odot \overrightarrow{w_2} = \sum_i c_i^{w_1} c_i^{w_2} \overrightarrow{n_i}$$

- An element-wise "mixture" of the input elements:



Tensor-based models

- One step further: Relational words are multi-linear maps (tensors of various orders) that can be applied to one or more arguments (vectors).



- The grammatical type of word determines the vector space in which the word lives

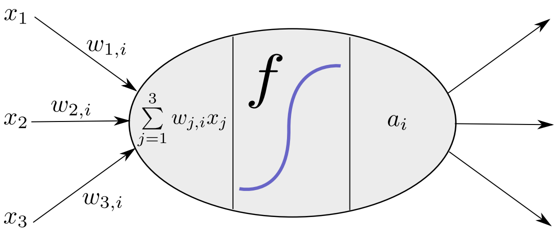
Baroni and Zamparelli (2010); Coecke, Sadrzadeh and Clark (2010)

From linear to non-linear models

- Composition in tensor-based models has to be a linear function.
- Linearity is an elegant property that allows for theoretical reasoning at a deep level:
- For example, Frobenius algebras have been used for modelling the meaning of relative pronouns and coordinators [Sadrzadeh et al. (2013); Kartsaklis (2016)]
- But word composition does not have to be linear; in fact, it has been shown that the application of consecutive non-linear transformations can be very effective in NLP tasks.
- Most of such neural models originated in image processing

Outline

An artificial neuron

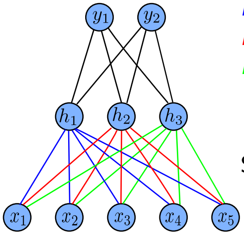


- The x_is form the input vector
- The w_{ji}s is a set of weights associated with the i-th output of the layer
- f is a non-linear function such as tanh or sigmoid
- a_i is the i-th output of the layer, computed as:

$$a_i = f(w_{1i}x_1 + w_{2i}x_2 + w_{3i}x_3)$$

A simple neural net

- A feed-forward neural network with one hidden layer:



$$h_1 = f(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + w_{41}x_4 + w_{51}x_5 + b_1)$$
$$h_2 = f(w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + w_{42}x_4 + w_{52}x_5 + b_2)$$
$$h_3 = f(w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + w_{43}x_4 + w_{53}x_5 + b_3)$$

or $\overrightarrow{h} = f(\mathbf{W}^{(1)}\overrightarrow{x} + \overrightarrow{b}^{(1)})$

Similarly:

$$\overrightarrow{y} = f(\mathbf{W}^{(2)}\overrightarrow{h} + \overrightarrow{b}^{(2)})$$

- Note that $\mathbf{W}^{(1)} \in \mathbb{R}^{3\times5}$ and $\mathbf{W}^{(2)} \in \mathbb{R}^{2\times3}$
- f is a non-linear function such as tanh or sigmoid
- A universal approximator

Objective functions (1/2)

- The goal of NN training is to find the set of parameters that optimizes a given objective function
- Or, to put it differently, to minimize an error function.
- Assume, for example, the goal of the NN is to produce a vector $\overrightarrow{\hat{y}}$ that matches an observed target vector $\overrightarrow{y}$. The function:

$$E = \frac{1}{n}\sum_{i=1}^{n} ||\overrightarrow{\hat{y}_i} - \overrightarrow{y_i}||^2$$

gives the total mean squared error across all training instances (useful in regression problems)

Objective functions (2/2)

- There are many different objective functions, and choosing the right one depends on the underlying task.
- For classification tasks, one would use cross-entropy:

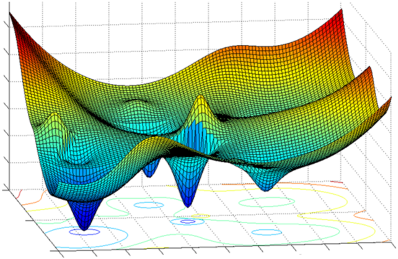
$$E = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i)$$

- Many other possibilities and variations exist.

In all cases, we want to set the weights of the NN such that E becomes zero or as close to zero as possible.

Gradient descent

- Initialize the parameters Θ randomly.
- Take steps proportional to the *negative* of the gradient of E at the current point.



$$\Theta_t = \Theta_{t-1} - \alpha \nabla E(\Theta_{t-1})$$

- Θ_t: the parameters of the model at time step t
- α: a learning rate

(Graph taken from "The Beginner Programmer" blog, http://firsttimeprogrammer.blogspot.co.uk)

Backpropagation of errors

- How do we compute the error terms at the inner layers?

These are computed based one the errors of the next layer by using backpropagation. In general:

$$\delta_k = \Theta_k^{\mathsf{T}} \delta_{k+1} \odot f'(z_k)$$

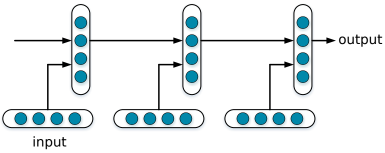
- δ_k is the error vector at layer k
- Θ_k is the weight matrix of layer k
- z_k is the weighted sum at the output of layer k
- f' is the derivative of the non-linear function f

- Just an application of the chain rule for derivatives.

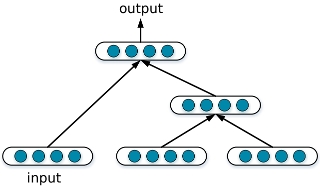
Outline

Recurrent and recursive NNs

- Standard NNs assume that inputs are independent of each other
- That is not the case in language; a word, for example, always depends on the previous words in the same sentence
- In a recurrent NN, connections form a directed cycle so that each output depends on the previous ones
- A recursive NN is applied recursively following a specific structure.



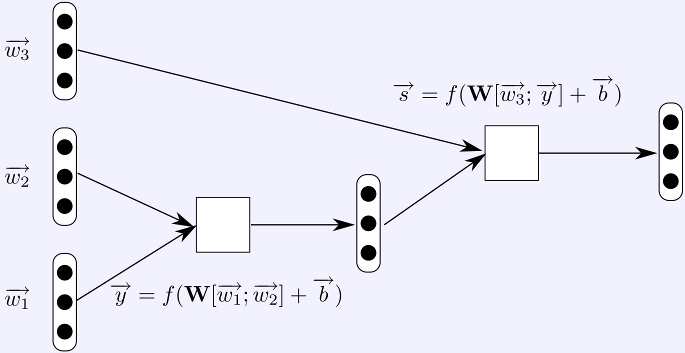
Recurrent NN



Recursive NN

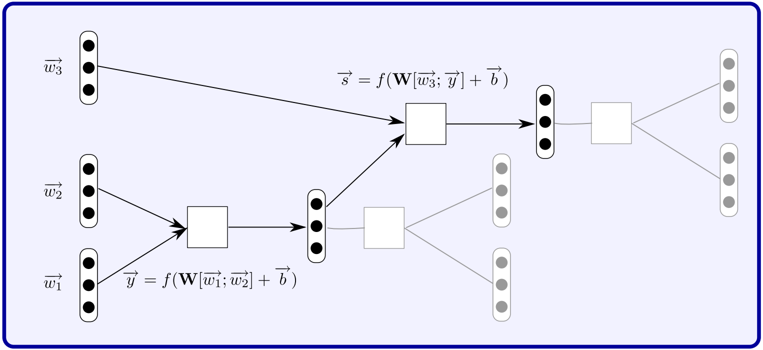
Recursive neural networks for composition

Pollack (1990); Socher et al. (2011;2012):



Unsupervised learning with NNs

- How can we train a NN in an unsupervised manner?
- Train the network to reproduce its input via an expansion layer:



- Use the output of the hidden layer as a compressed version of the input [Socher et al. (2011)]

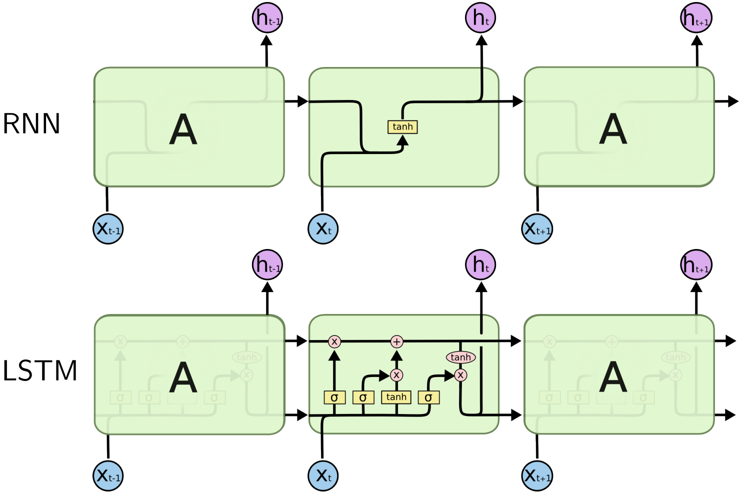
Long Short-Term Memory networks (1/2)

- RNNs are effective, but fail to capture long-range dependencies such as:

> **The movie I** liked and John said Mary and Ann really **hated**

- "Vanishing gradient" problem: Back-propagating the error requires the multiplication of many very small numbers together, and training for the bottom layers starts to stall.
- Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) provide a solution, by equipping each neuron with an internal state.

Long Short-Term Memory networks (2/2)



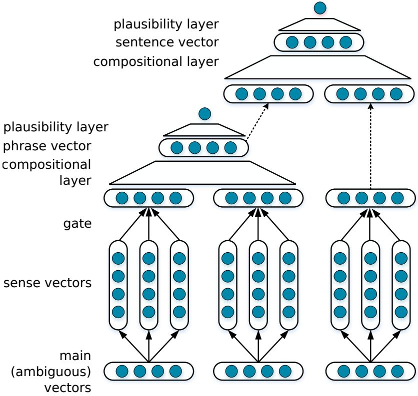
(Diagrams taken from Christopher Olah's blog, http://colah.github.io/)

Linguistically aware RNNs

NN-based methods come mainly from image processing. How can we make them more linguistically aware?

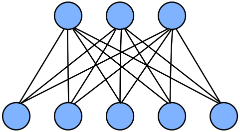
Cheng and Kartsaklis (2015):

- Take into account syntax, by optimizing against a scrambled version of each sentence
- Dynamically disambiguate the meaning of words during training based on their context

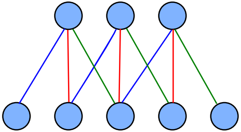


Convolutional NNs

- Originated in pattern recognition [Fukushima, 1980]
- Small filters apply on every position of the input vector:



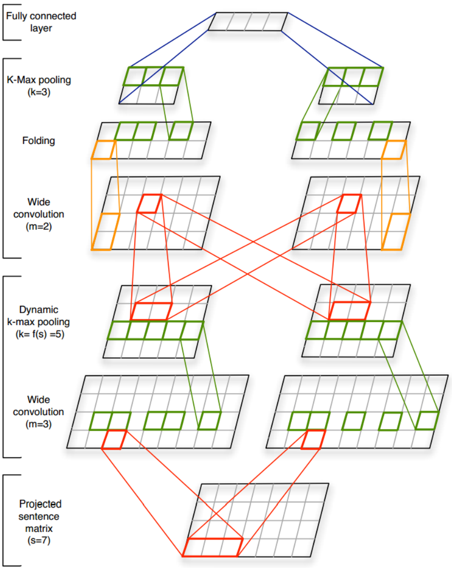
Fully-connected layer



Convolutional layer

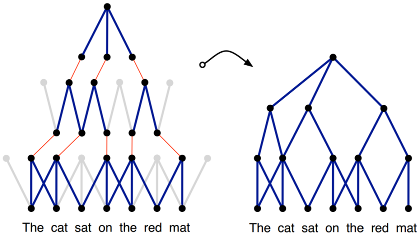
- Capable of extracting fine-grained local features independently of the exact position in input
- Features become increasingly global as more layers are stacked
- Each convolutional layer is usually followed by a pooling layer
- Top layer is fully connected, usually a soft-max classifier
- Application to language: Collobert and Weston (2008)

DCNNs for modelling sentences



Kalchbrenner, Grefenstette and Blunsom (2014): A deep architecture using dynamic k-max pooling

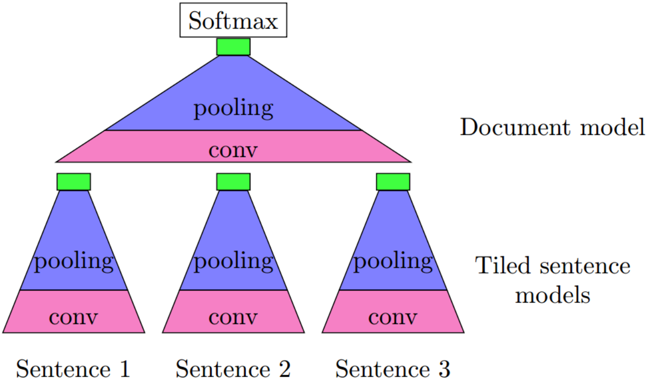
- Syntactic structure is induced automatically:



(*Figures reused with permission*)

Beyond sentence level

An additional convolutional layer can provide document vectors [Denil et al. (2014)]:



(*Figure reused with permission*)

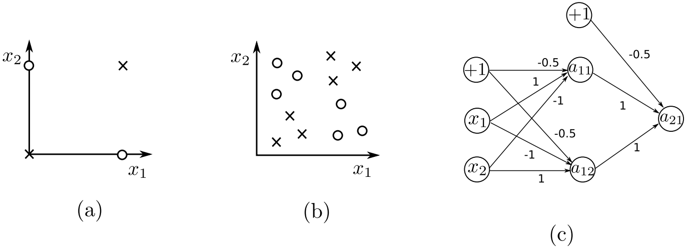
Neural models: Intuition (1/2)

- Recall that tensor-based composition involves a linear transformation of the input into some output.
- Neural models make this process more effective by applying consecutive non-linear layers of transformation.

A NN does not only project a noun vector onto a sentence space, but it can also transform the geometry of the space itself in order to make it reflect better the relationships between the points (sentences) in it.

Neural models: Intuition (2/2)

- **Example:** Although there is no linear map to send an input $x \in \{0, 1\}$ to the correct XOR value, the function can be approximated by a simple NN with one hidden layer.



- Points in (b) can be seen as representing two semantically distinct groups of sentences, which the NN is able to distinguish (while a linear map cannot)

Outline

Neural models: Pros and Cons

Distinguishing feature:

Drastic transformation of the sentence space.

PROS:

- Non-linearity and layered approach allow the simulation of a very wide range of functions
- Word vectors are parameters of the model, optimized during training
- State-of-the-art results in a number of NLP tasks

CONS:

- Requires expensive training based on backpropagation
- Difficult to discover the right configuration
- A "black-box" approach: not easy to correlate inner workings with output

A final word...

We should feel excited and glad to live in a time when NLP is seen as so central to both the further development of machine learning and industry application problems. However, I would encourage everyone to think about problems, architectures, cognitive science, and the details of human language, how it is learned, processed, and how it changes, rather than just chasing state-of-the-art numbers on a benchmark task.

—Chris Manning,
Computational Linguistics, 41:4

Thank you for your attention!

References I

Baroni, M. and Zamparelli, R. (2010).
Nouns are Vectors, Adjectives are Matrices.
In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Cheng, J. and Kartsaklis, D. (2015).
Syntax-aware multi-sense word embeddings for deep compositional models of meaning.
In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal. Association for Computational Linguistics.

Coecke, B., Sadrzadeh, M., and Clark, S. (2010).
Mathematical Foundations for a Compositional Distributional Model of Meaning. Lambek Festschrift.
Linguistic Analysis, 36:345–384.

Collobert, R. and Weston, J. (2008).
A unified architecture for natural language processing: Deep neural networks with multitask learning.
In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Denil, M., Demiraj, A., Kalchbrenner, N., Blunsom, P., and de Freitas, N. (2014).
Modelling, visualising and summarising documents with a single convolutional neural network.
Technical Report arXiv:1406.3830, University of Oxford.

Harris, Z. (1968).
Mathematical Structures of Language.
Wiley.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014).
A convolutional neural network for modelling sentences.
In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

References II

Kartsaklis, D. (2016).
Coordination in Categorical Compositional Distributional Semantics.
In *Proceedings of the 2016 Workshop on Semantic Spaces at the Intersection of NLP, Physics and Cognitive Science*. EPTCS.

Mitchell, J. and Lapata, M. (2010).
Composition in distributional models of semantics.
Cognitive Science, 34(8):1388–1439.

Sadrzadeh, M., Clark, S., and Coecke, B. (2013).
The Frobenius anatomy of word meanings I: subject and object relative pronouns.
Journal of Logic and Computation, Advance Access.

Sadrzadeh, M., Clark, S., and Coecke, B. (2014).
The Frobenius anatomy of word meanings II: Possessive relative pronouns.
Journal of Logic and Computation.

Socher, R., Huang, E., Pennington, J., Ng, A., and Manning, C. (2011).
Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection.
Advances in Neural Information Processing Systems, 24.

Socher, R., Huval, B., Manning, C., and A., N. (2012).
Semantic compositionality through recursive matrix-vector spaces.
In *Conference on Empirical Methods in Natural Language Processing 2012*.