Generating Natural-Language Text with Neural Networks

Jonathan Mugan
@jmugan

Data Day Texas
January 27, 2018

DeUmbra

Overview of this talk on neural text generation

Me: co-founder and CEO at DeepGrammar; I am also a principal scientist at DeUmbra. We do work for the DoD.

This talk will cover: writing text using neural networks, like a chatbot or webpage generator

- Advantage: text can be created from data without painful hand-coding of templates
  - Imagine being able to create text customized by user personality
- Disadvantage: currently hard to control
  - E.g., the vectors for “Tuesday” and “Wednesday” are similar

We will see a sequence of abstractions:

seq2seq models →
encoder-decoder models →
generative text models →
generative behavior models

DeUmbra

Outline

- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

Outline

- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

DeUmbra

Sequence-to-sequence models

A sequence-to-sequence (seq2seq) model

1. encodes a sequence of tokens, such as a sentence, into a single vector
2. decodes that vector into another sequences of tokens

Both the encoding and decoding are often done using recurrent neural networks (RNNs).

The initial big application for this was machine translation. For example, where the source sentences are English and the target sentences are Spanish.

Seq2seq works on all kinds of sentence pairs

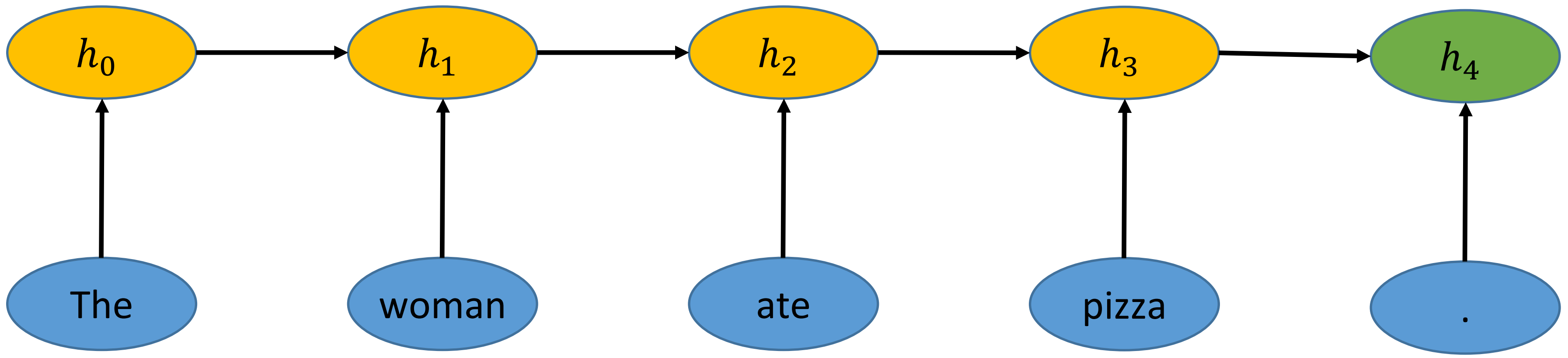
For example, take a bunch of dialogs and use machine learning to predict what the next statement will be given the last statement.

- Movie and TV subtitles
  - OpenSubtitles http://opus.lingfil.uu.se/OpenSubtitles.php
- Can mine Twitter
  - Look for replies to tweets using the API
- Ubuntu Dialog Corpus
  - Dialogs on people wanting technical support https://arxiv.org/abs/1506.08909

Another example, text summarization. E.g., learn to generate headlines for news articles. See https://memkite.com/deeplearningkit/2016/04/23/deep-learning-for-text-summarization/

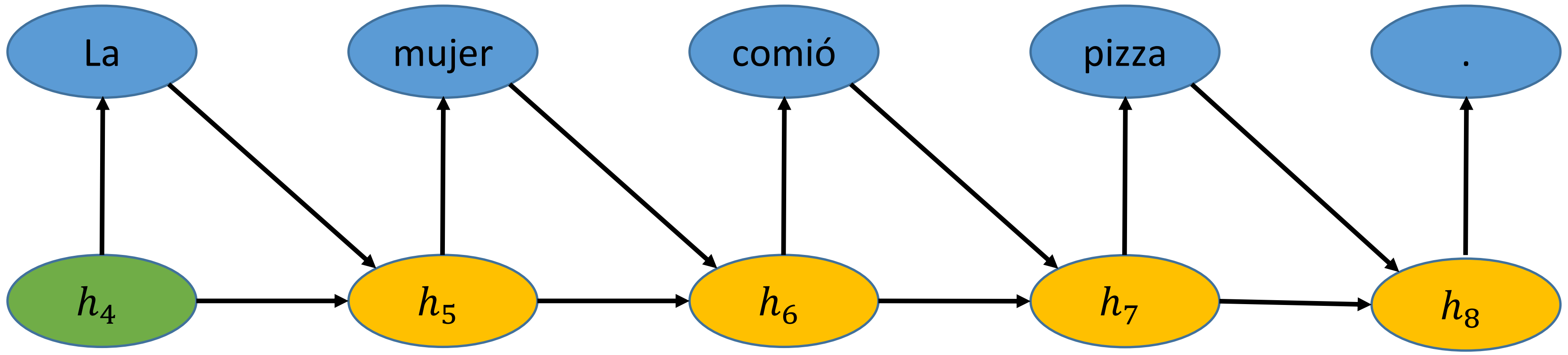
Encoding a sentence for machine translation

As each word is examined, the encoder RNN integrates it into the hidden state vector h.



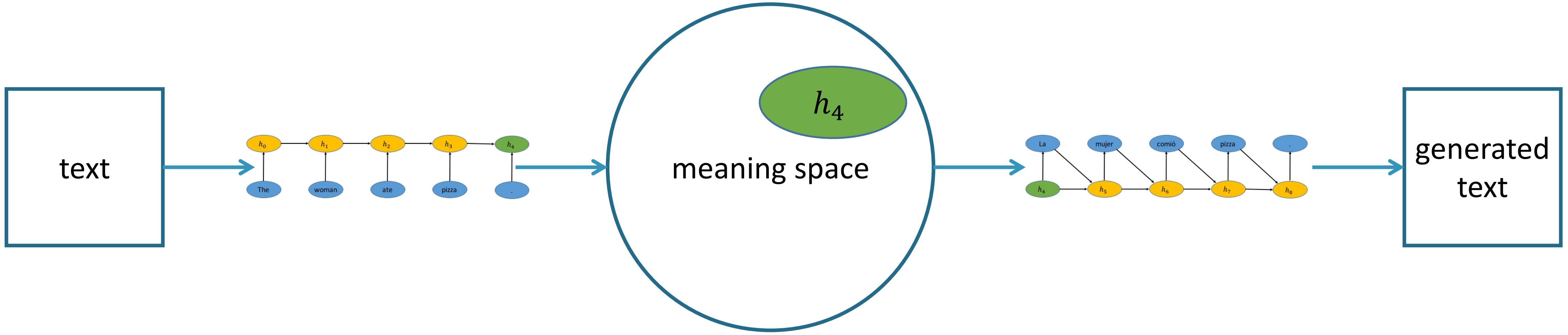
Decoding the sentence into Spanish

The model begins with the last hidden state from the encoder and generates words until a special stop symbol is generated.



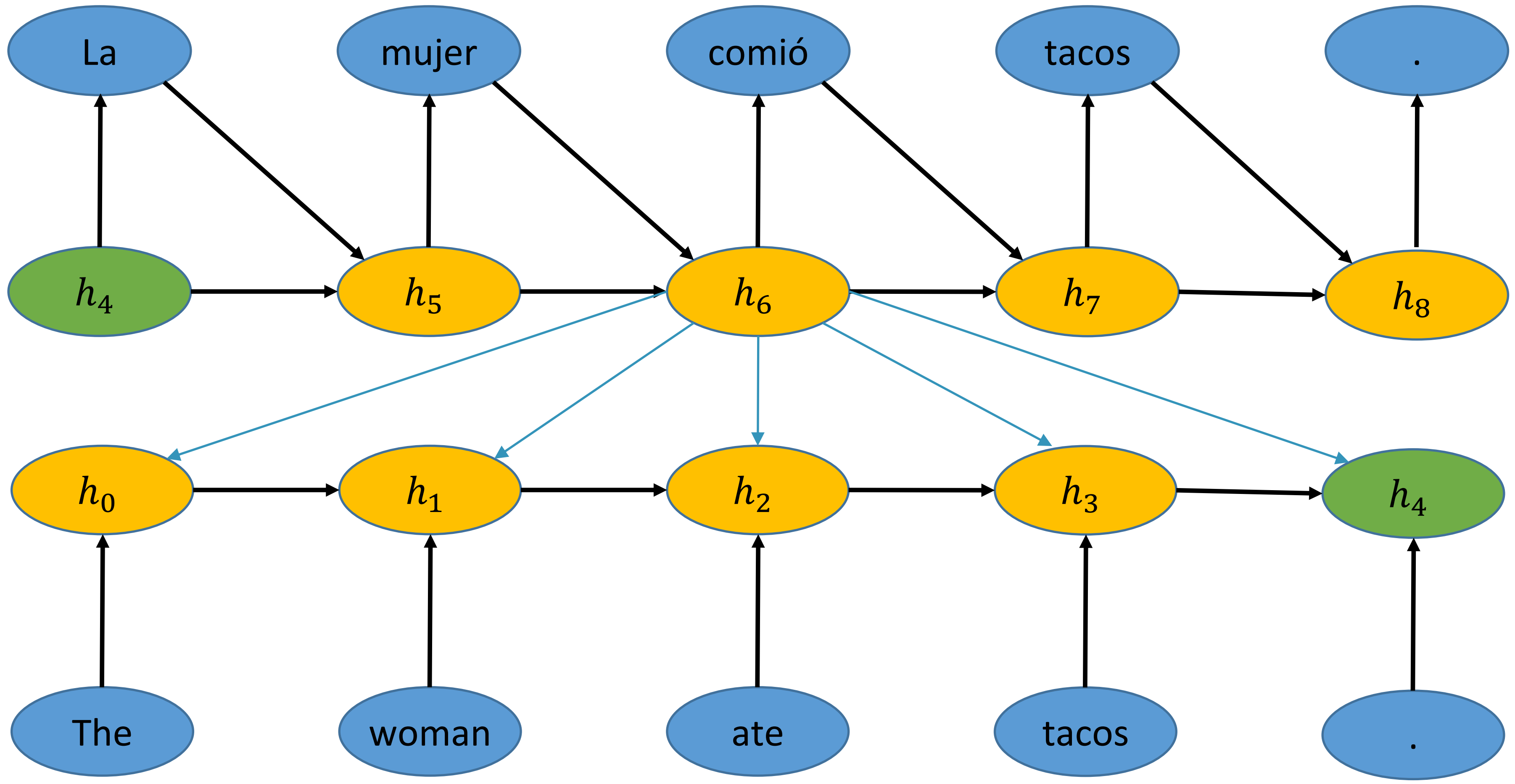
DeUmbra

Generalized view



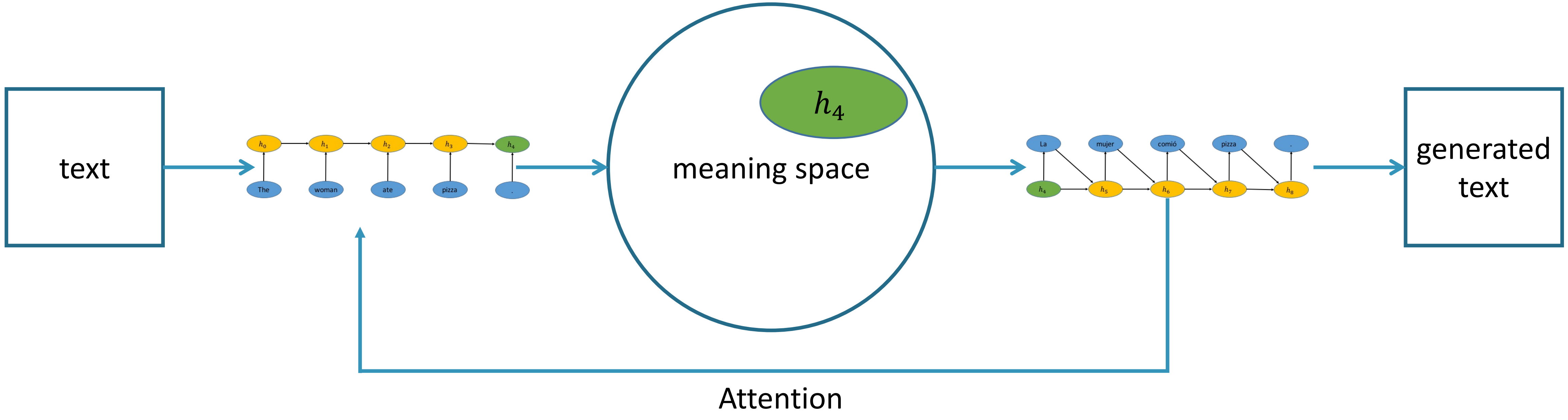
Attention helps the model consider more information

The network learns how to determine which previous memory is most relevant at each decision point [Bahdanau et al., 2014].

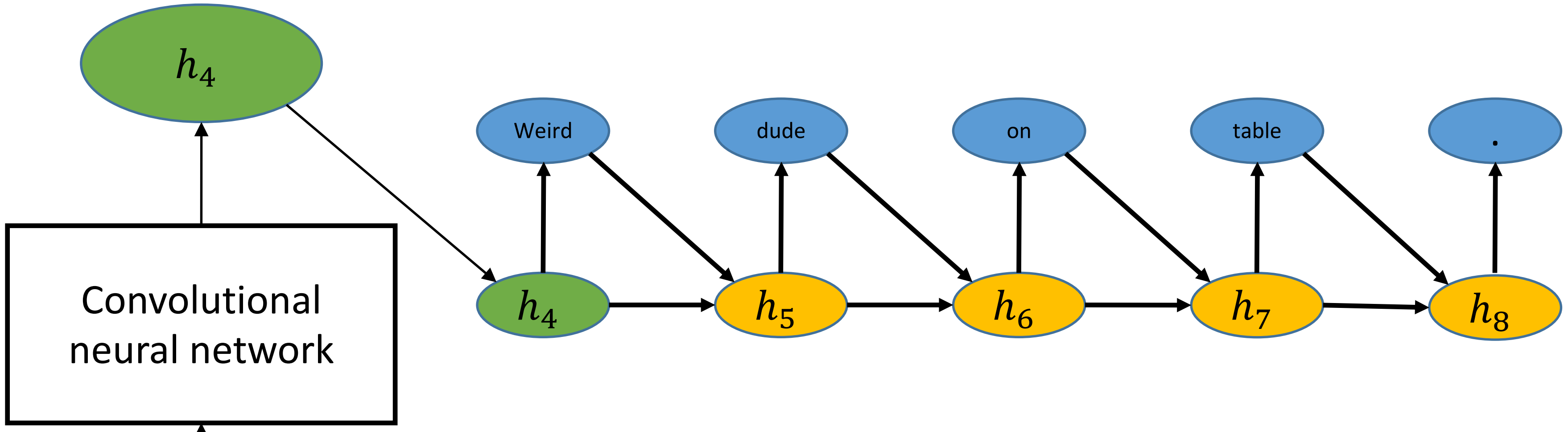


DeUmbra

RNN seq2seq model with attention



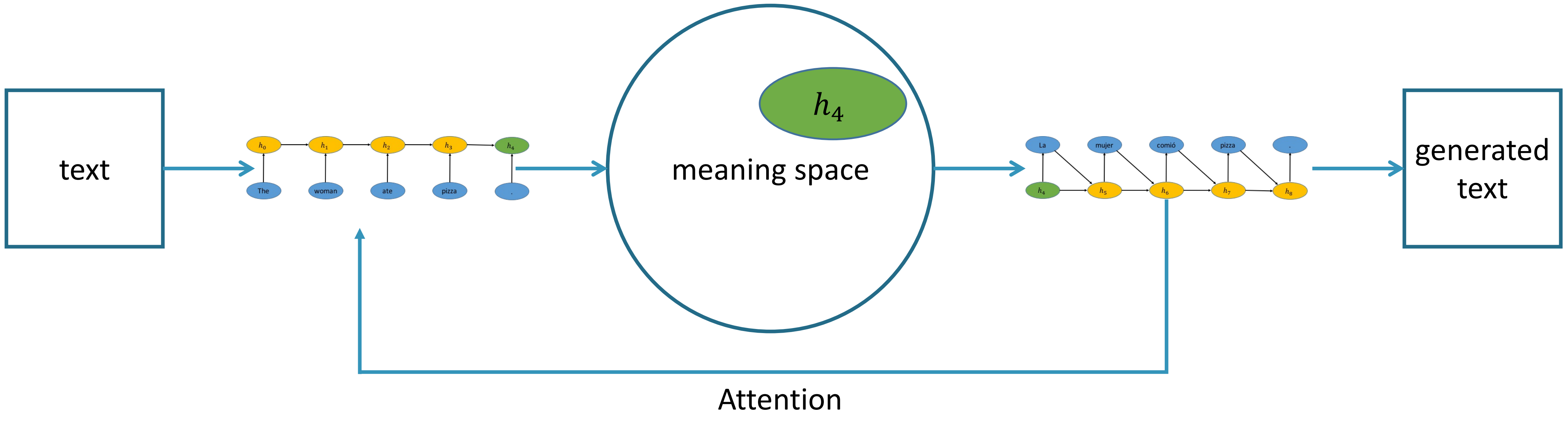
Seq2seq can be broadened to encoder-decoder



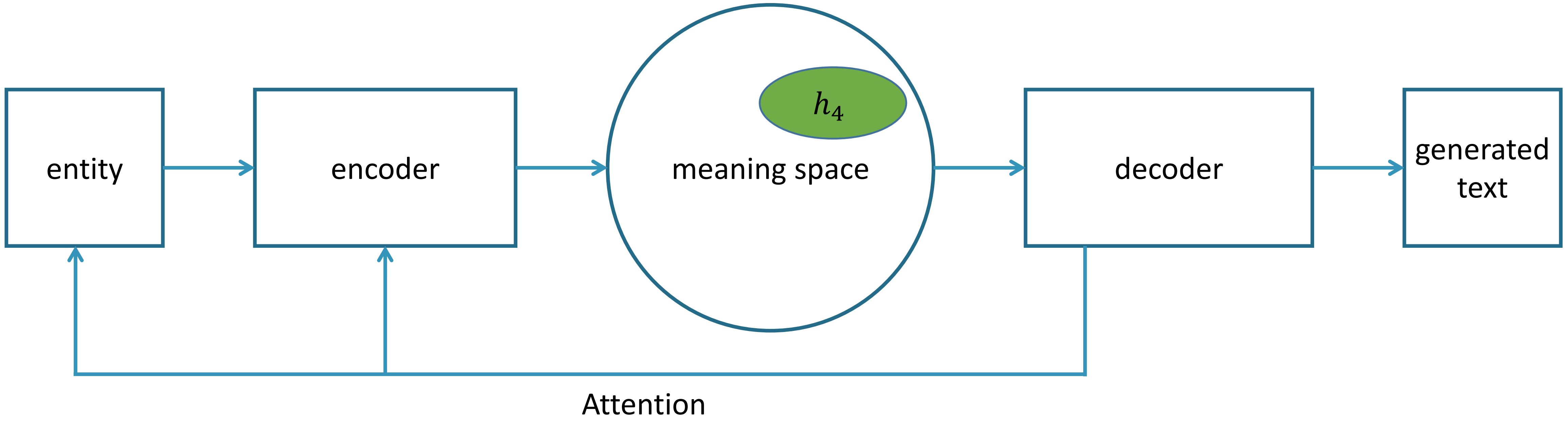
Karpathy and Fei-Fei, 2015
http://cs.stanford.edu/people/karpathy/deepimagesent/

Vinyals et al., 2014
http://googleresearch.blogspot.com/2014/11/a-picture-is-worth-thousand-coherent.html

RNN seq2seq model with attention



Generalized encoder-decoder model family



Outline

- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

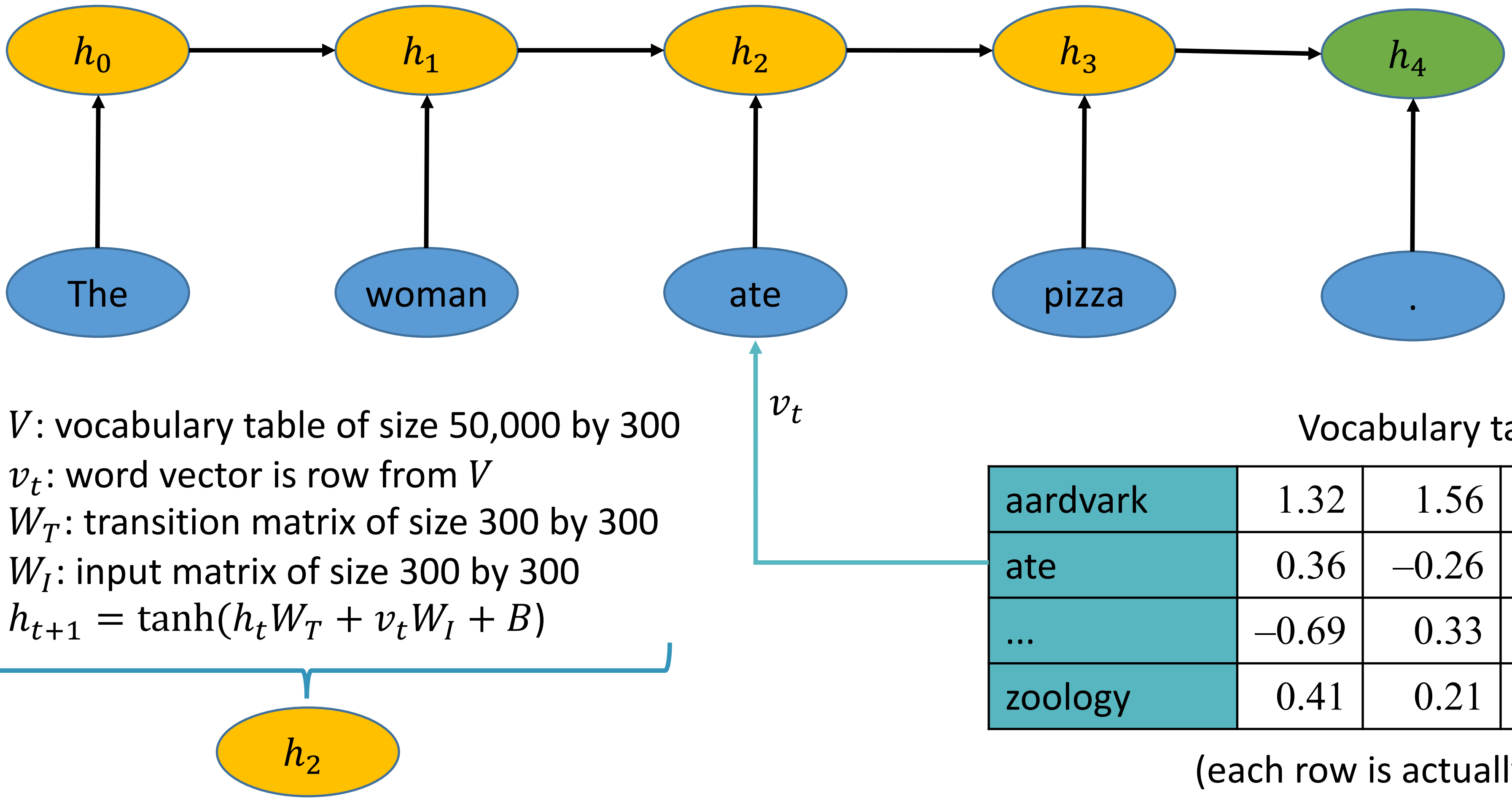
DeUmbra

Concrete example of an RNN encoder

Let's look at a simple cell so we can get a grounded understanding.

See Christopher Olah's blog post for great presentation of more complex cells like LSTMs

http://colah.github.io/posts/2015-08-Understanding-LSTMs/



Parameters start off as random values and are updated through backpropagation.

- V: vocabulary table of size 50,000 by 300
- v_t: word vector is row from V
- W_T: transition matrix of size 300 by 300
- W_I: input matrix of size 300 by 300
- $h_{t+1} = \tanh(h_t W_T + v_t W_I + B)$

$$\underbrace{\hspace{8em}}_{h_2}$$

Vocabulary table

aardvark	1.32	1.56	0.31	–1.21	0.31
ate	0.36	–0.26	0.31	–1.99	0.11
...	–0.69	0.33	0.77	0.22	–1.29
zoology	0.41	0.21	–0.32	0.31	0.22

(each row is actually 300 dimensions)

CNN Explanation

- 128 filters of width 3, 4, and 5
- Total of 384 filters

0.21	0.56	0.21
1.12	–1.12	1.12
0.55	0.26	0.55
0.81	1.71	–0.31
–0.66	9.66	0.66

1) For each word in the sentence, get its word vector from the vocabulary table.

Tom	ate	with	his	brother
0.23	0.36	0.23	0.23	0.23
1.42	–0.26	1.32	1.27	–1.42
–0.33	0.31	0.33	0.59	0.33
1.23	–1.99	1.23	–1.31	1.23
1.11	0.11	0.11	2.19	1.83

- 2) Slide each filter over the sentence and element-wise multiply and add (dot product); then take the max of the values seen.
- Result is a sentence vector of length 384 that represents the sentence.

- For classification: each class has a vector of weights of length 384
- 3) To decide which class, the class vector is multiplied element wise by the sentence vector and summed (dot product)
  - The class with the highest sum wins

Vocabulary table

aardvark	1.32	1.56	0.31	–1.21	0.31
ate	0.36	–0.26	0.31	–1.99	0.11
...	–0.69	0.33	0.77	0.22	–1.29
zoology	0.41	0.21	–0.32	0.31	0.22

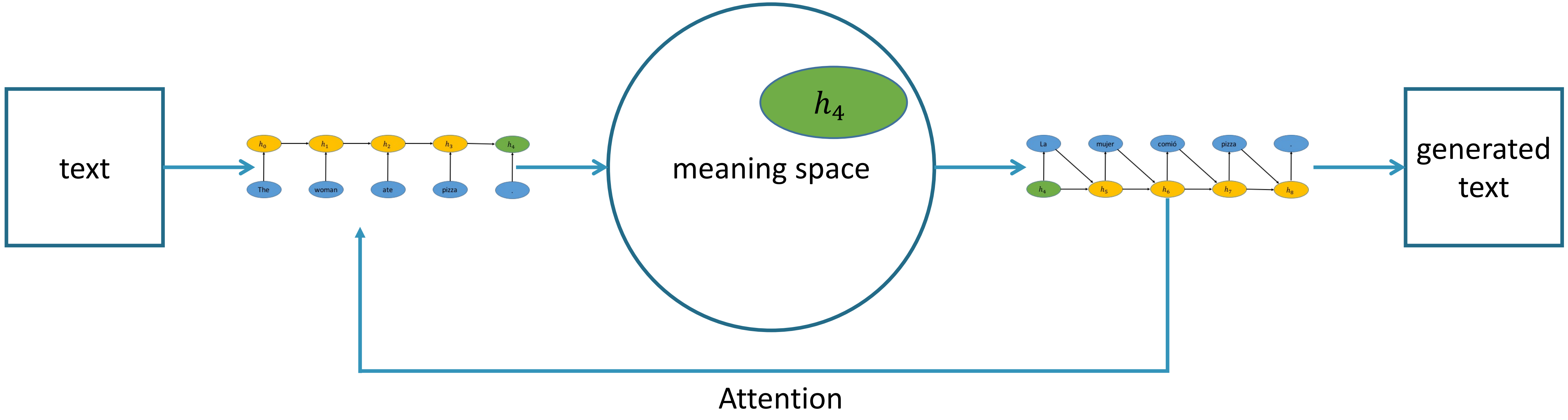
(each row is actually 300 dimensions)

- See the code here by Denny Britz https://github.com/dennybritz/cnn-text-classification-tf
- README.md contains reference to his blog post and the paper by Y. Kim

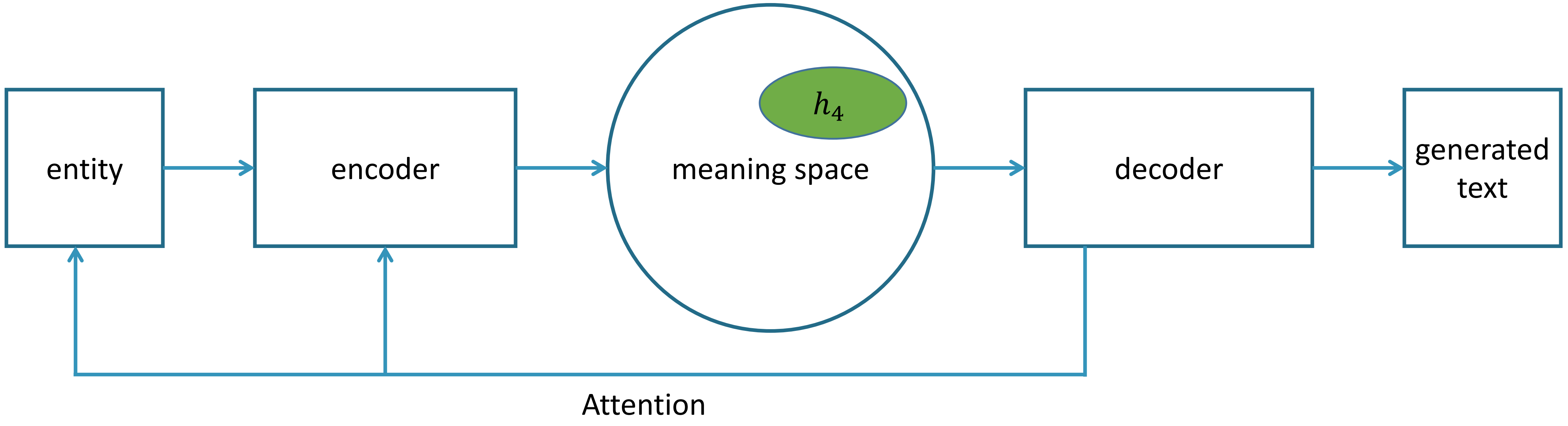
Outline

- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

RNN seq2seq model with attention



Generalized encoder-decoder model family



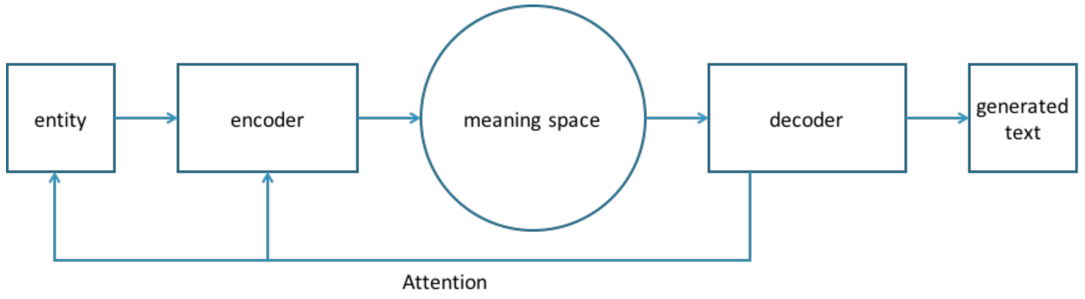
- We want to abstract to have a generative model of text
- This will allow us to move beyond translation into broader text generation

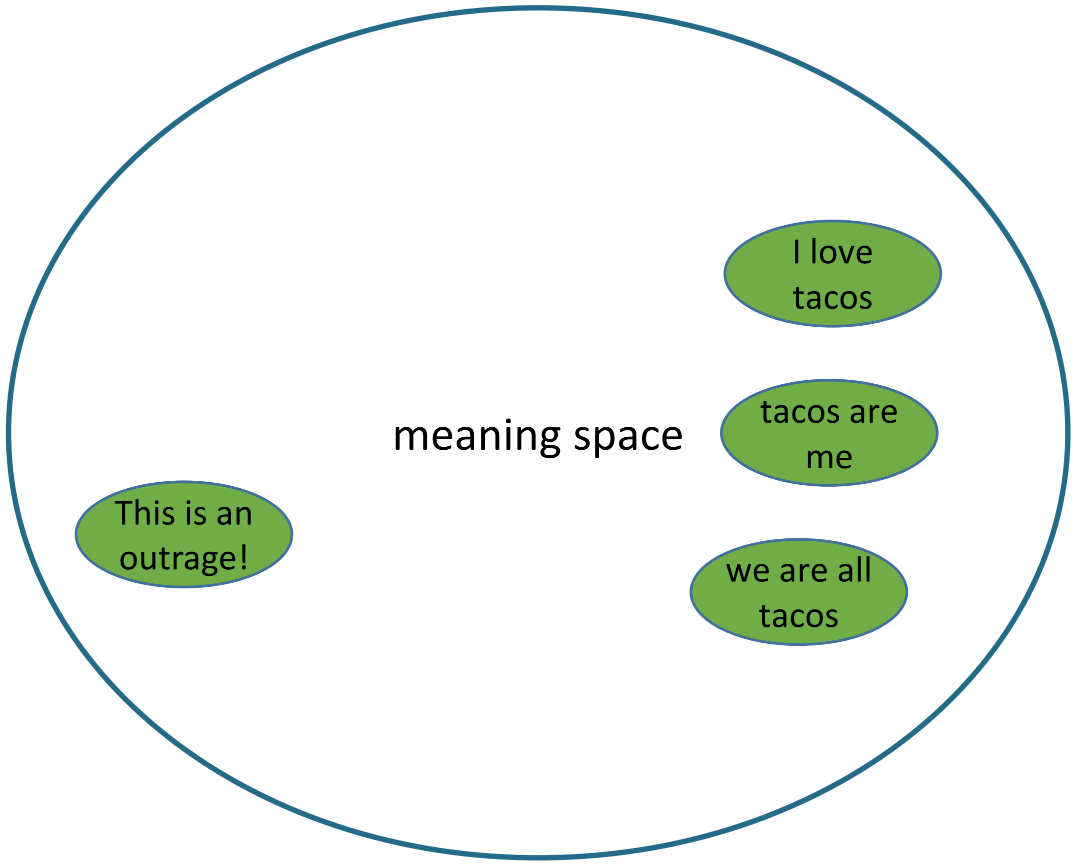
Generative models represent the probability of data

$$p(x) = \sum_i p(x|h_i)p(h_i)$$

The meaning space should represent the latent factors that result in the text.

Generalized view



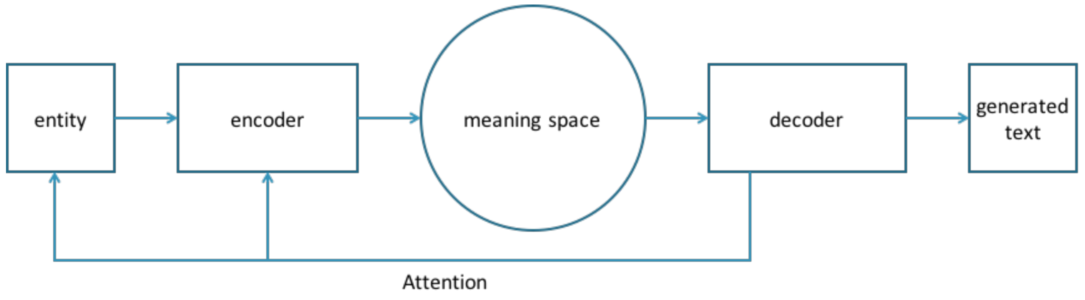


- We want to abstract to have a generative model of text
- This will allow us to move beyond translation into broader text generation

- You want semantically meaningful things to be near each other.
- You want to be able to interpolate between sentences.

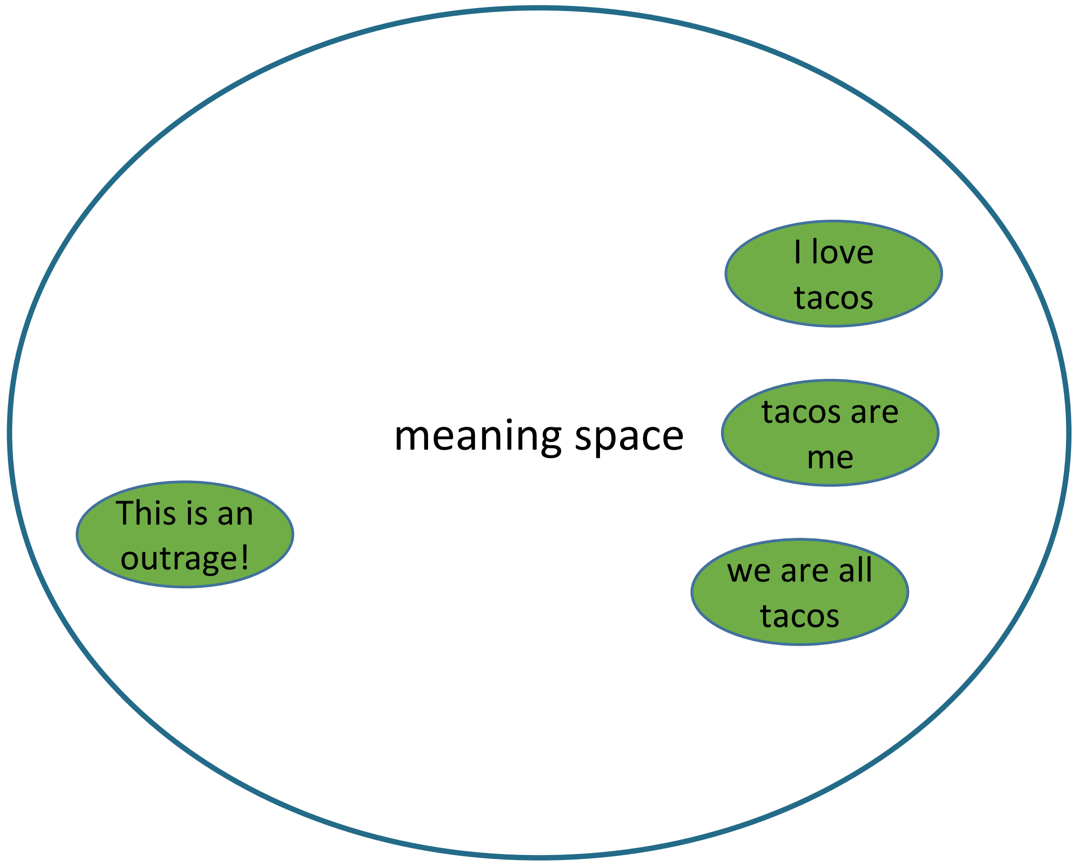
DeUmbra

Variational methods provide structure to meaning space



Variational methods can provide this structure.

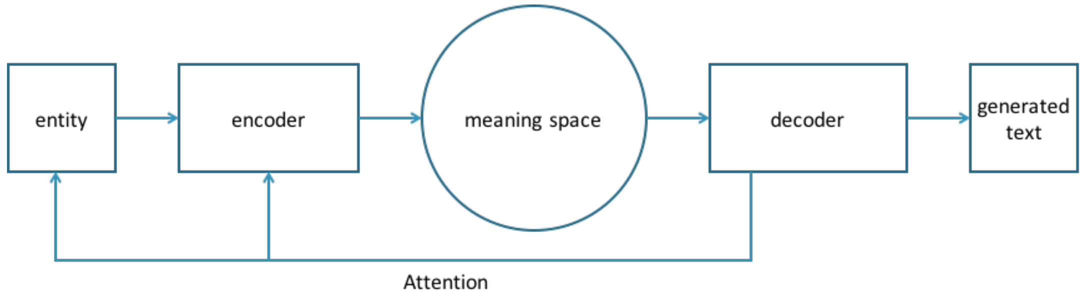
- In the movie *Arrival*, variational principles in physics allowed one to know the future.
- In our world, variational comes from the calculus of variations; optimization over a space of functions
- Variational inference is coming up with a simpler function that closely approximates the true function.



- ○ You want semantically meaningful things to be near each other.
- ○ You want to be able to interpolate between sentences.

DeUmbra

Variational methods provide structure to meaning space



Our simpler function for $p(h|x)$ is $q(h|x) = N(\mu, \sigma)$

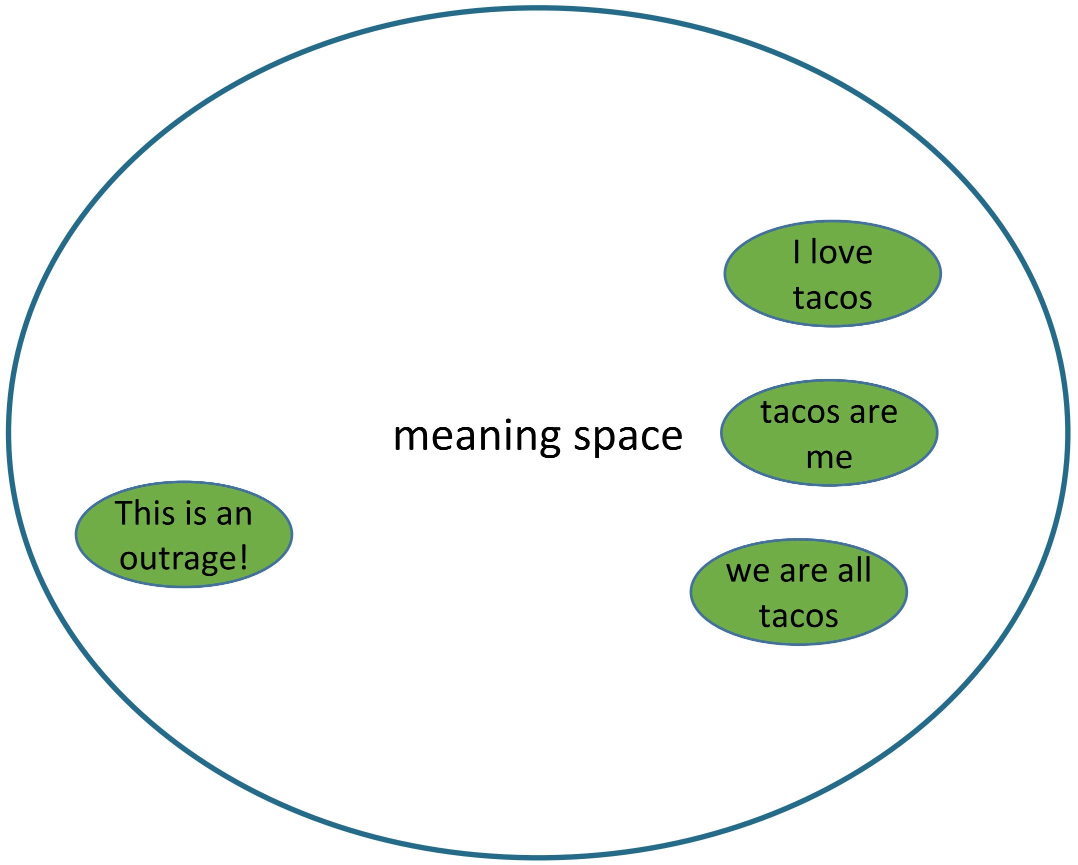
We use a neural network (RNN) to output μ and σ

To maximize this approximation, it turns out that we maximize the evidence lower bound (ELBO):

$$\mathcal{L}(x, q) = E_{q(h|x)}[\log p(x|h)] - KL(q(h|x)||p(h))$$

In practice, we only take one sample and we use the reparameterization trick to keep it differentiable.

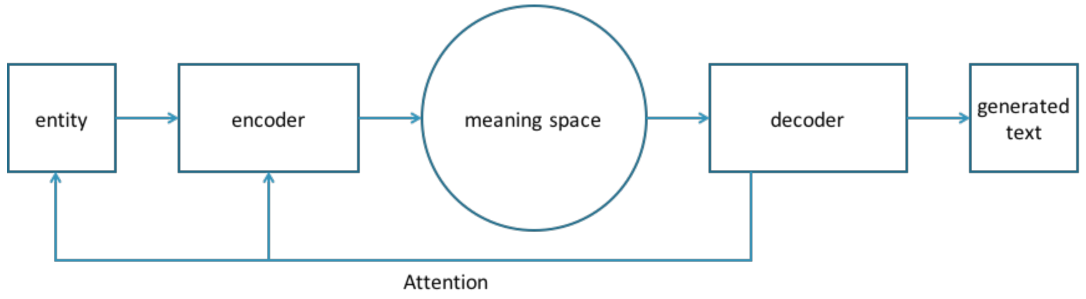
See Kingma and Welling https://arxiv.org/pdf/1312.6114.pdf



- You want semantically meaningful things to be near each other.
- You want to be able to interpolate between sentences.

DeUmbra

Variational methods provide structure to meaning space



Note that the equation:

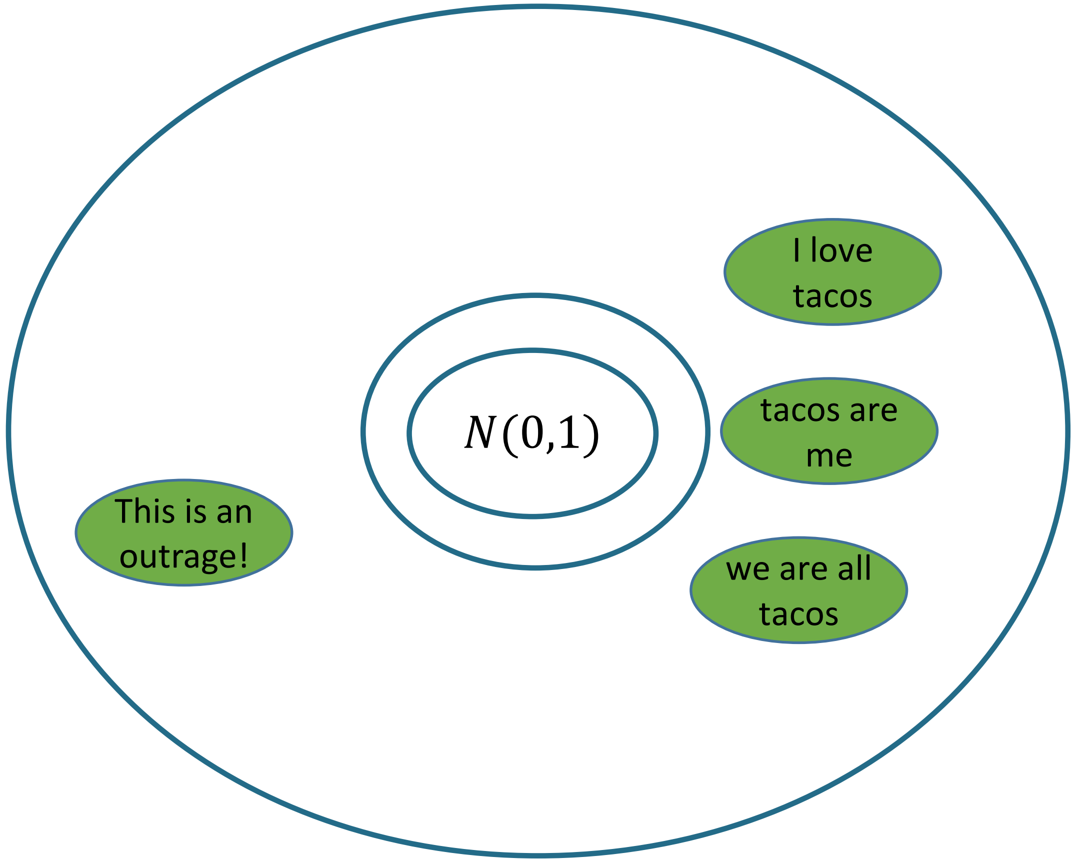
$$\mathcal{L}(x,q) = E_{q(h|x)}[\log p(x|h)] - KL(q(h|x)||p(h))$$

has the prior $p(h)$

The prior $p(h) = N(0,I)$ forces the algorithm to use the space efficiently, which forces it to put semantically similar latent representations together. Without this prior, the algorithm could put h values anywhere in the latent space that it wanted.

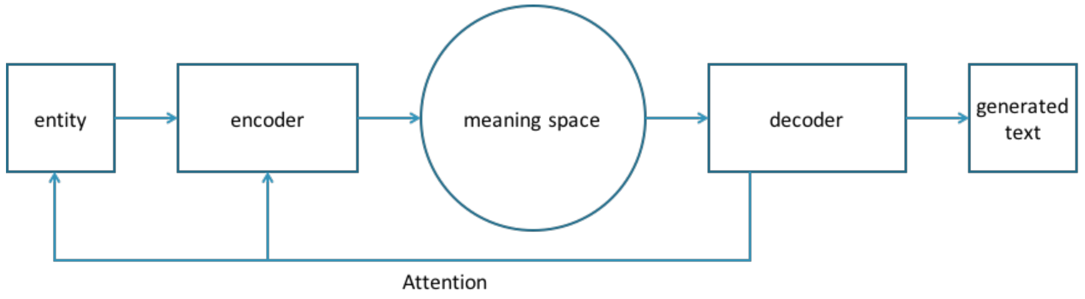
The prior acts like regularization, so we are maximizing the reconstruction fidelity $p(x|h)$ and minimizing the difference of h from 0.

Can be tricky to balance these things. Use KL annealing, see Bowman et al., https://arxiv.org/pdf/1511.06349.pdf



- You want semantically meaningful things to be near each other.
- You want to be able to interpolate between sentences.

We can directly manipulate the meaning space to generate desired text



Hu et al. allow one to manipulate single values of the meaning vector to change what is written
https://arxiv.org/abs/1703.00955

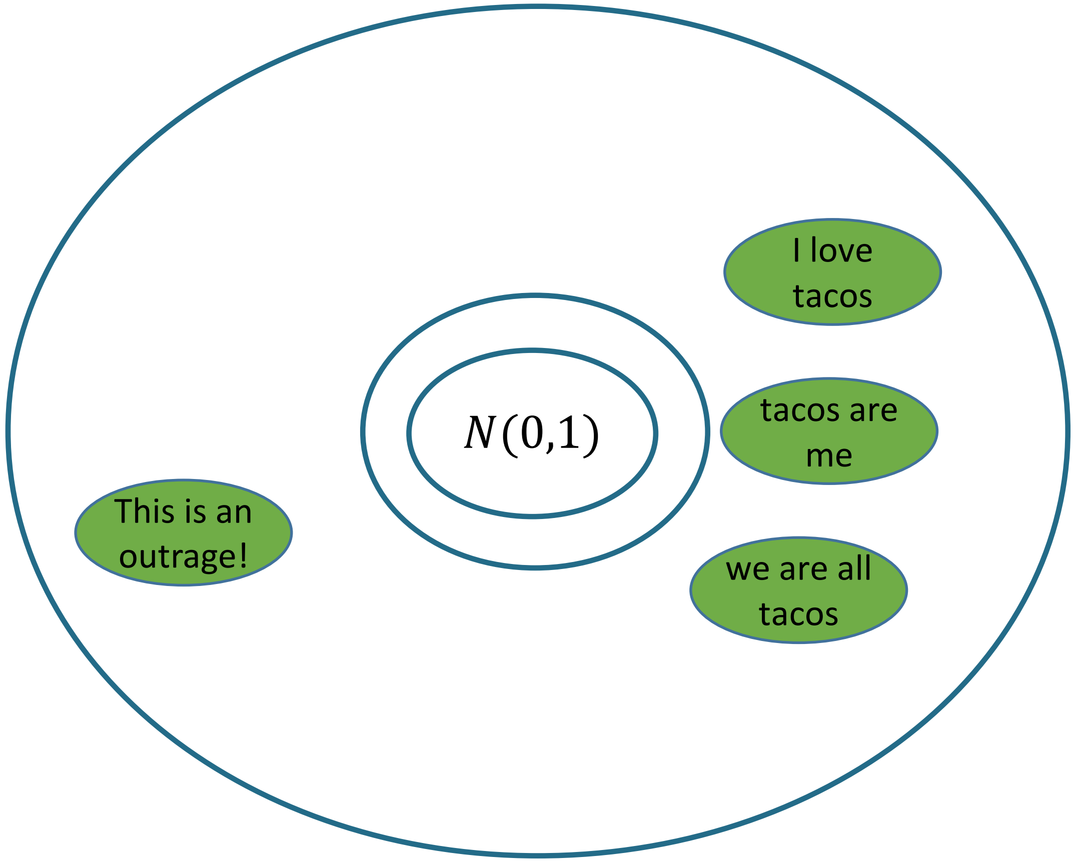
Examples on movie reviews:

(“negative”, “past”)
the acting was also kind of hit or miss .

(“positive”, “past”)
his acting was impeccable

(“negative”, “present”)
the movie is very close to the show in plot and characters

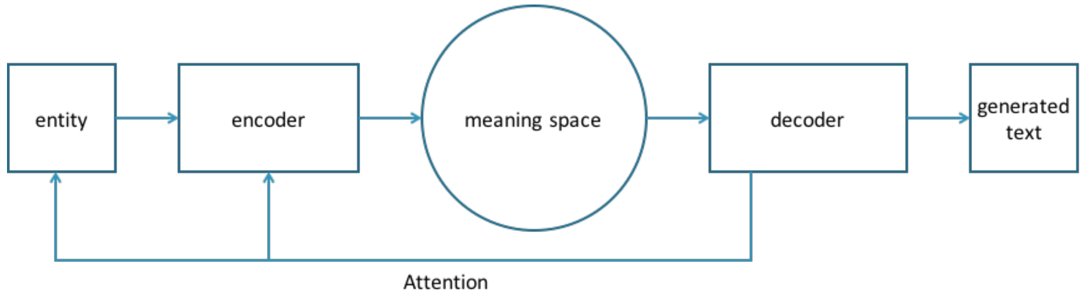
(“positive”, “present”)
this is one of the better dance films



In Hu et al., the disentangled representation comes from a discriminator that we can consider to be part of the encoder.

DeUmbra

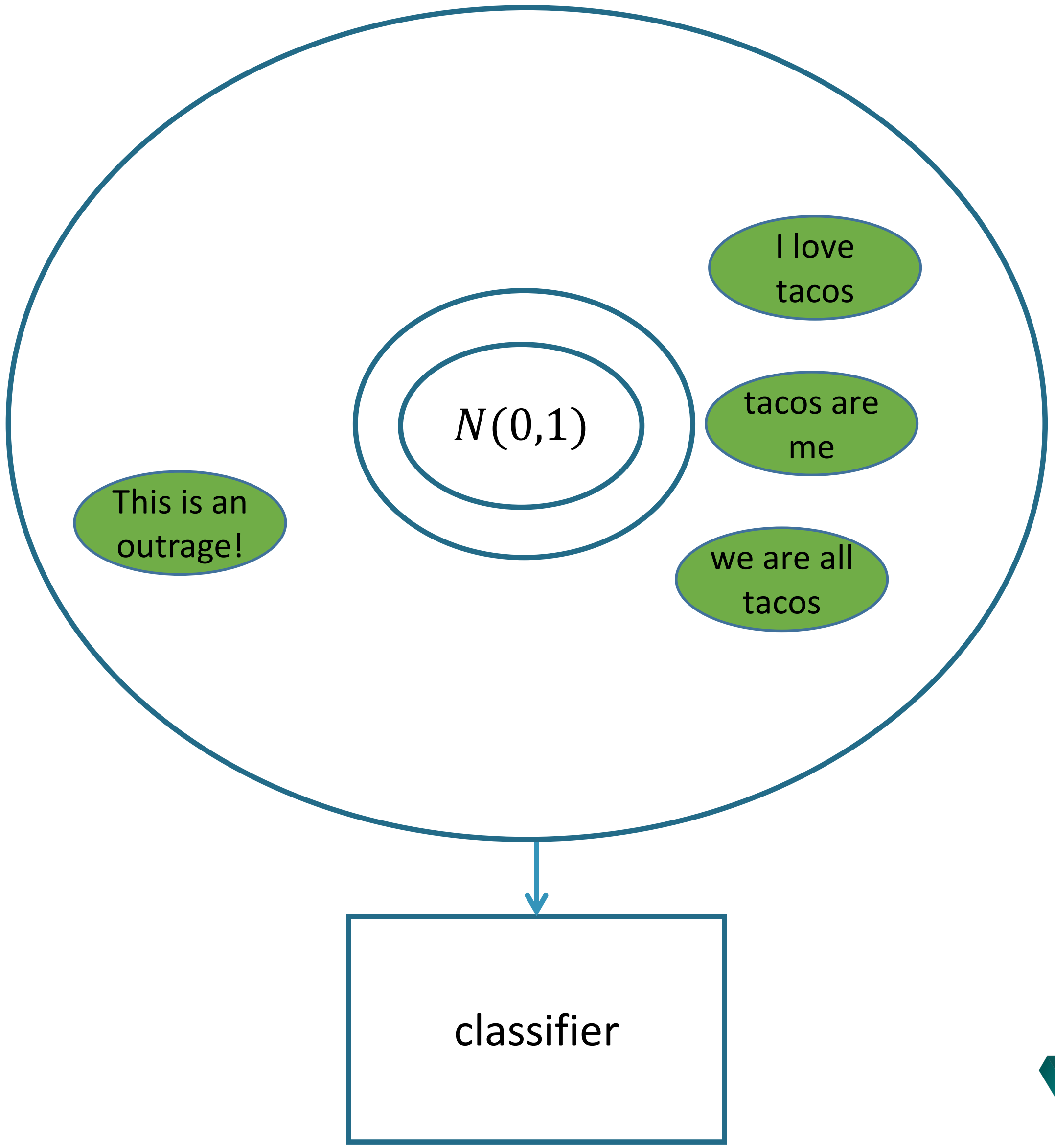
We can also do classification over meaning space



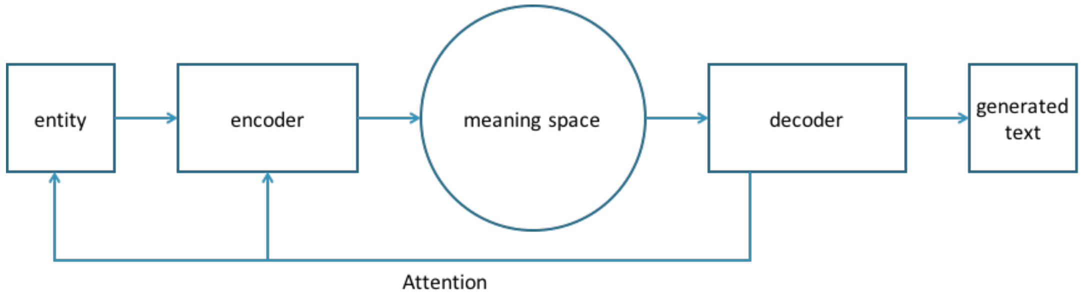
When we train with variational methods to regenerate the input text, this is called a variational autoencoder.

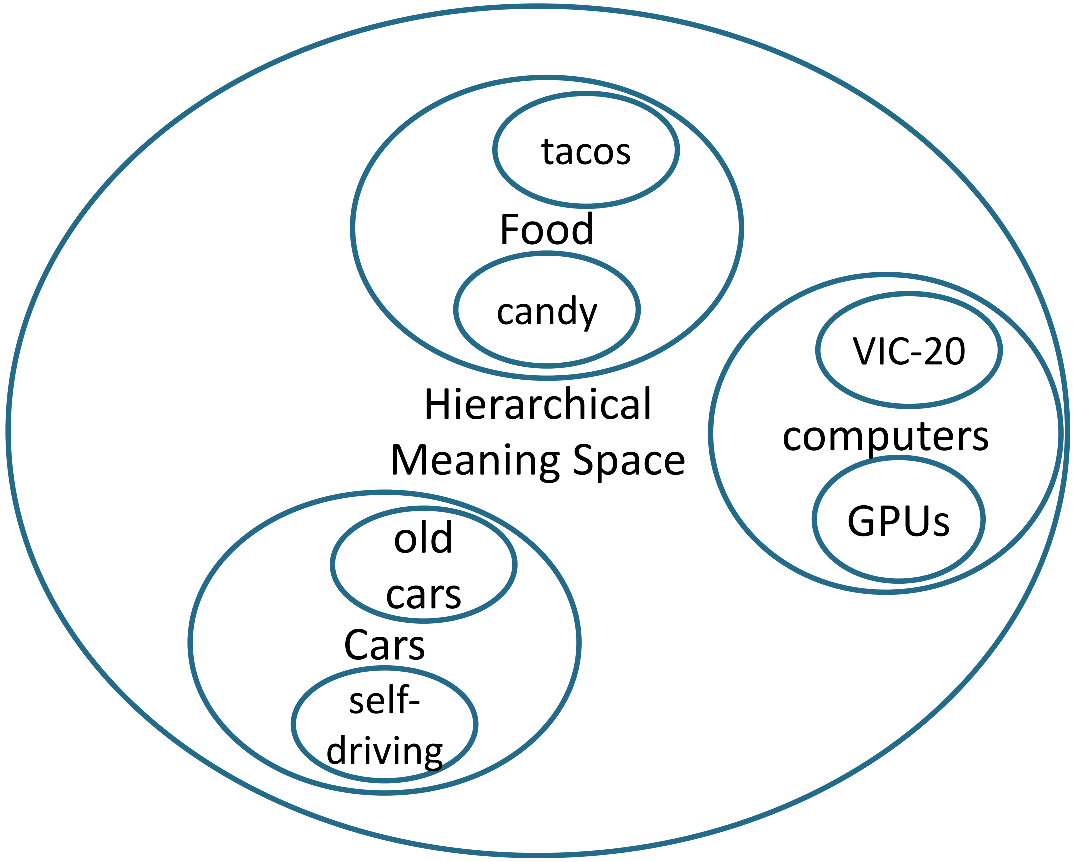
Since the space of things that needs to be represented is bigger than the number of parameters, the system is forced to learn to distill the essence. See nice OpenAI blog post https://blog.openai.com/generative-models/

This can be the basis for learning with less labeled training data (semi-supervised learning). See Kingma et al., https://arxiv.org/pdf/1406.5298.pdf



We can place a hierarchical structure on the meaning space





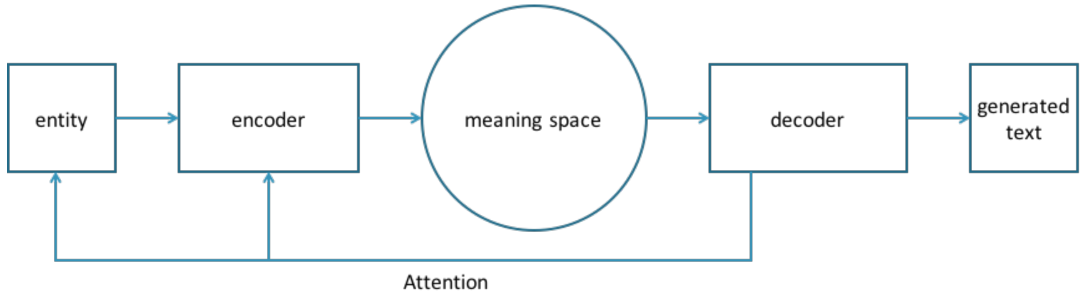
In the vision domain, Goyal et al. allows one to learn a hierarchy on the prior space using a hierarchical Dirichlet process
https://arxiv.org/pdf/1703.07027.pdf

The Dirichlet process has allows for an infinite number of clusters. As the world changes and new concepts emerge, you can keep adding clusters.

Videos on understanding Dirichlet processes and the related Chinese restaurant processes by Tamara Broderick
part I https://www.youtube.com/watch?v=kKZkNUvsJ4M
part II https://www.youtube.com/watch?v=oPcv8MaESGY
part III https://www.youtube.com/watch?v=HpcGlr19vNk

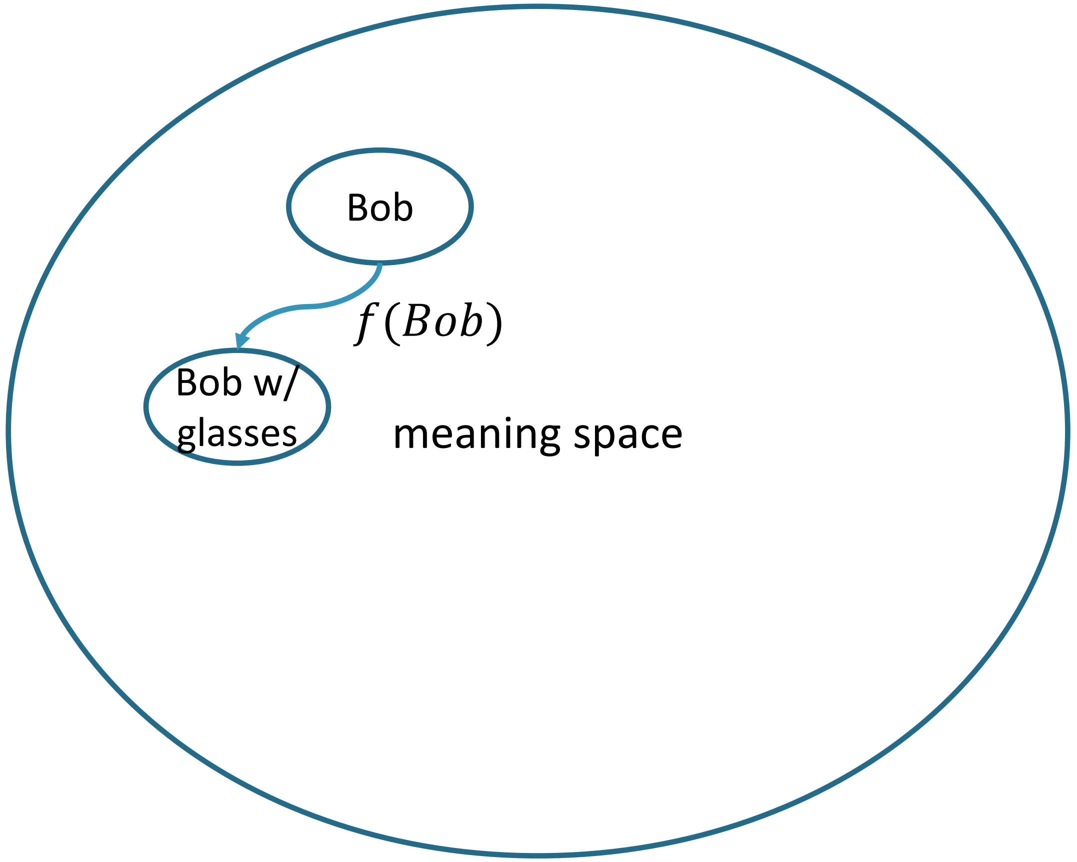
DeUmbra

We can learn functions to find desired parts of the meaning space



In the vision domain, Engel et al. allows one to translate a point in meaning space to a new point with desired characteristics
https://arxiv.org/pdf/1711.05772.pdf

In general, these kinds of things work better in vision than with text because pixel values are smoother than discrete word choice. But, we will see how people are working to overcome this restriction.



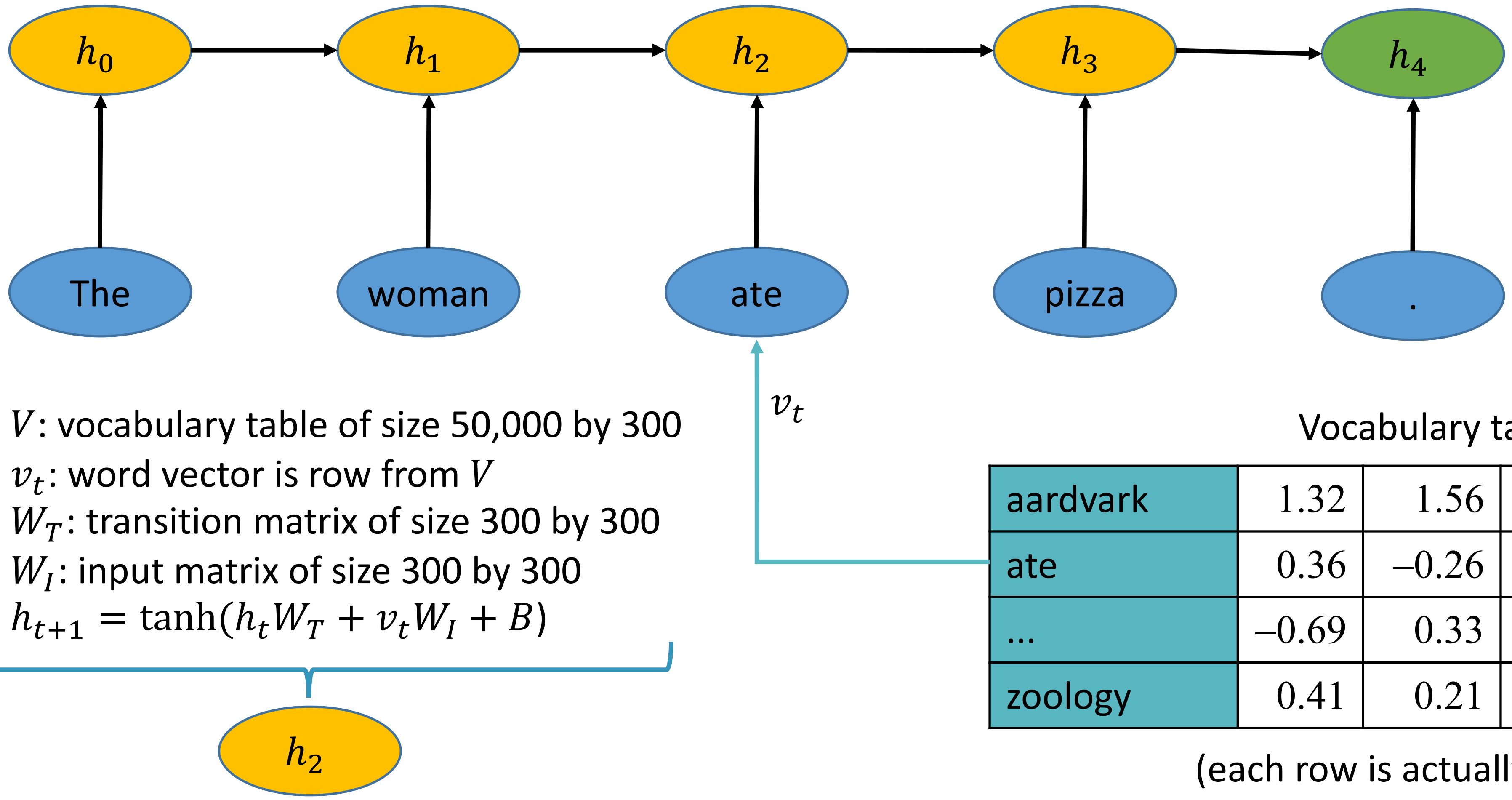
Interesting idea: the meaning space can be a meaning space of programs
[see Liang et al., https://arxiv.org/pdf/1611.00020.pdf and Yang et al., https://arxiv.org/pdf/1711.06744.pdf]

DeUmbra

Outline

- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

Recall our RNN encoder



- V: vocabulary table of size 50,000 by 300
- v_t: word vector is row from V
- W_T: transition matrix of size 300 by 300
- W_I: input matrix of size 300 by 300
- $h_{t+1} = \tanh(h_t W_T + v_t W_I + B)$

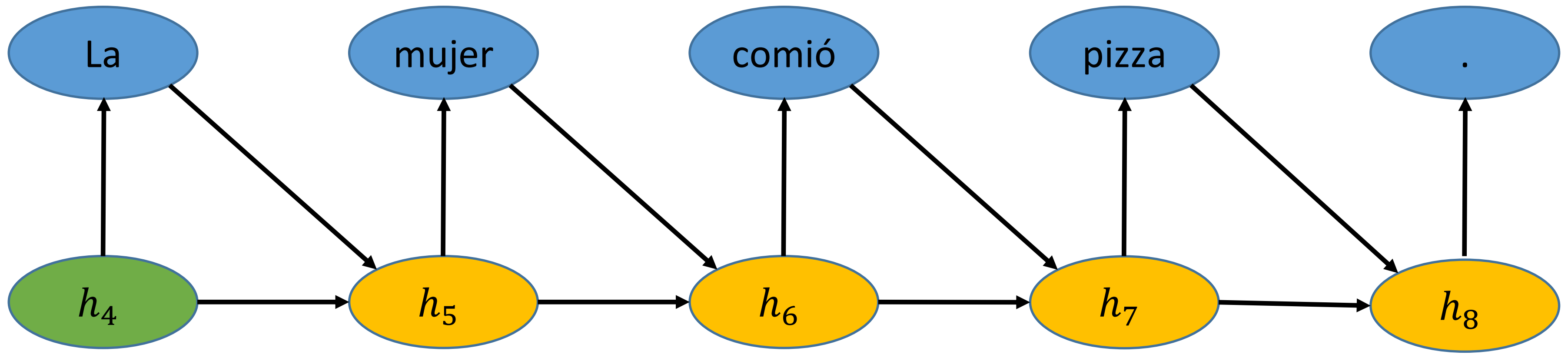
Vocabulary table

aardvark	1.32	1.56	0.31	–1.21	0.31
ate	0.36	–0.26	0.31	–1.99	0.11
…	–0.69	0.33	0.77	0.22	–1.29
zoology	0.41	0.21	–0.32	0.31	0.22

(each row is actually 300 dimensions)

DeUmbra

Concrete example of RNN decoder

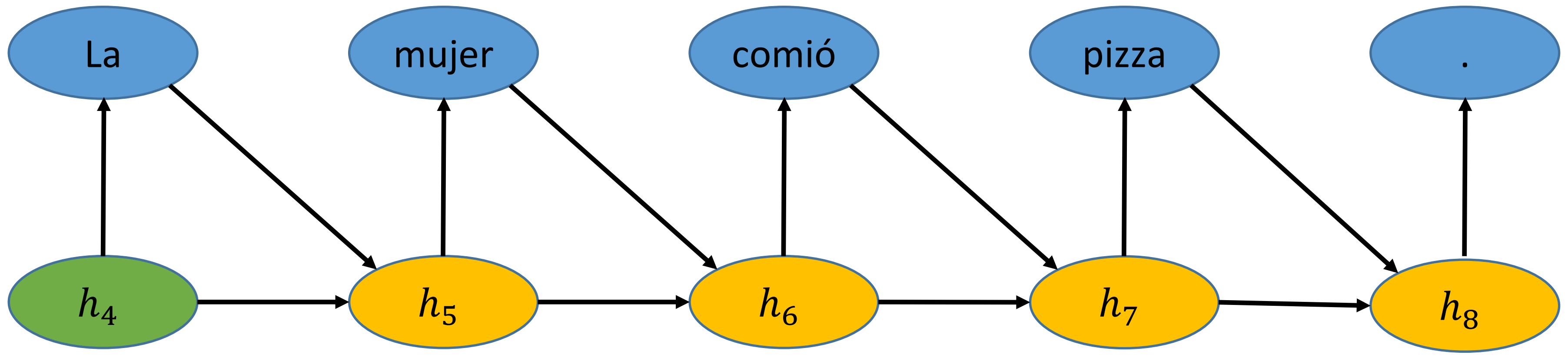


- V: vocabulary table of size 50,000 by 300
- v_{t-1}: word vector is row from V, v_5 is "mujer"
- W_T^D: transition matrix of size 300 by 300
- W_I^D: input matrix of size 300 by 300
- $\underbrace{h_t = \tanh(h_{t-1}W_T^D + v_{t-1}W_I^D + B)}_{h_6}$

- W_O^D: output matrix of size 300 by 50,000
- A probability distribution over next words $p(words) = softmax(h_t W_O^D)$
- The softmax function makes them all sum to 1 $softmax(x_j) = e^{x_j} / \sum_i e^{x_i}$

DeUmbra

Models are trained with paired sentences to minimize cost



At time $t = 6$ we know the correct word is "comió", so the cost is

$$cost = -\log(p(\text{comió}))$$

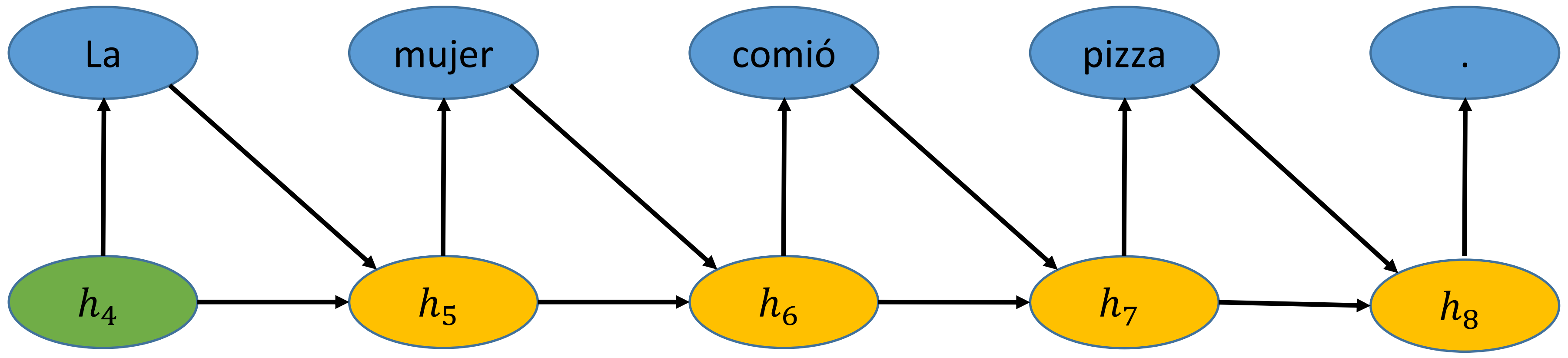
So the more likely "comió" is, the lower the cost.

Also, note that regardless of what the model said at time $t-1$, we feed in "mujer" at time t. This is called *teacher forcing.*

- W_O^D: output matrix of size 300 by 50,000
- A probability distribution over next words $p(words) = softmax(h_t W_O^D)$
- The softmax function makes them all sum to 1 $softmax(x_j) = e^{x_j} / \sum_i e^{x_i}$

DeUmbra

When it comes time to generate text



When you are actually using the model to generate text, you don't know what the right answer is.

too cheap
too expensive

You could do a greedy decoder
- Take the most likely first word, then given that word, takes the most likely next word, and so on.
- But that could lead you down a suboptimal path.

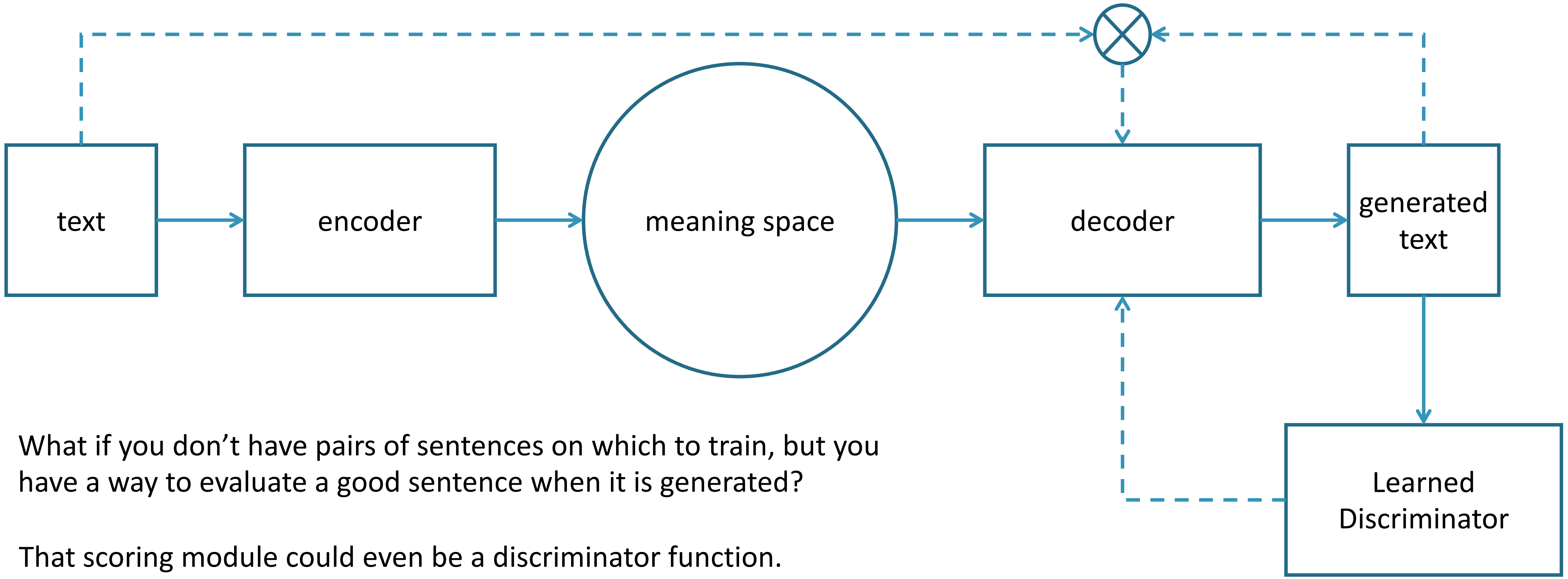
A compromise is beam search
- Start generating with the first word
- Keep the 10 most promising sequences of words

You generate look at all sequences of words up to a given length
- Choose the sequence with the lowest cost
- But there are far too many.

Beam search can sometimes lack diversity, and there have been proposals to address this;
see Li and Jurafsky, https://arxiv.org/pdf/1601.00372v2.pdf and Ashwin et al., https://arxiv.org/pdf/1610.02424v1.pdf

DeUmbra

Beyond pairs and teacher forcing



What if you don't have pairs of sentences on which to train, but you have a way to evaluate a good sentence when it is generated?

That scoring module could even be a discriminator function.
Then you have a Generative Adversarial Network (GAN).

If you can't do teacher forcing, you have to wait until the end, so how do you know if the word generated at time t was a good one?

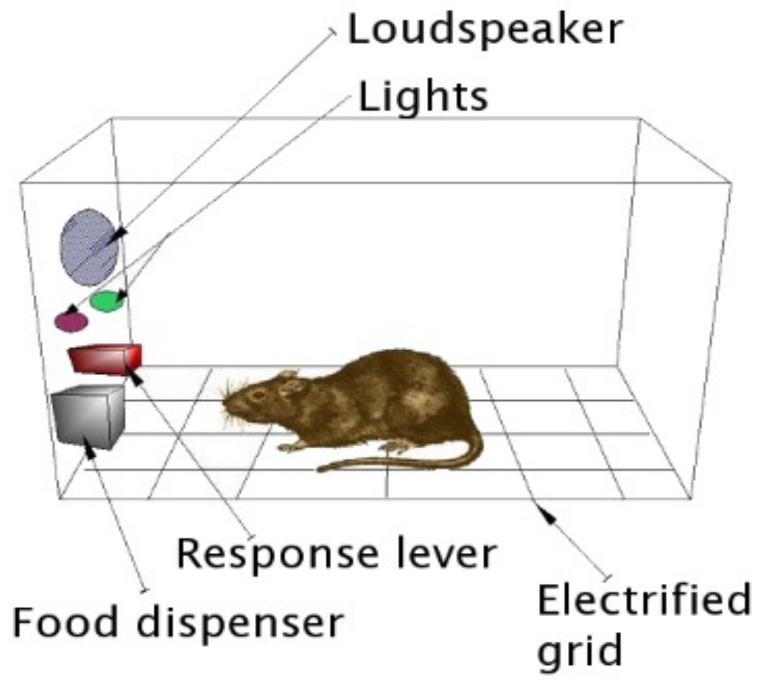
Generating text is not differentiable, so you have to use reinforcement learning.

- In GANs the discriminator is trained to tell real from fake text.
- But it could be any kind of evaluation function.

DeUmbra

Reinforcement learning is a gradual stamping in of behavior

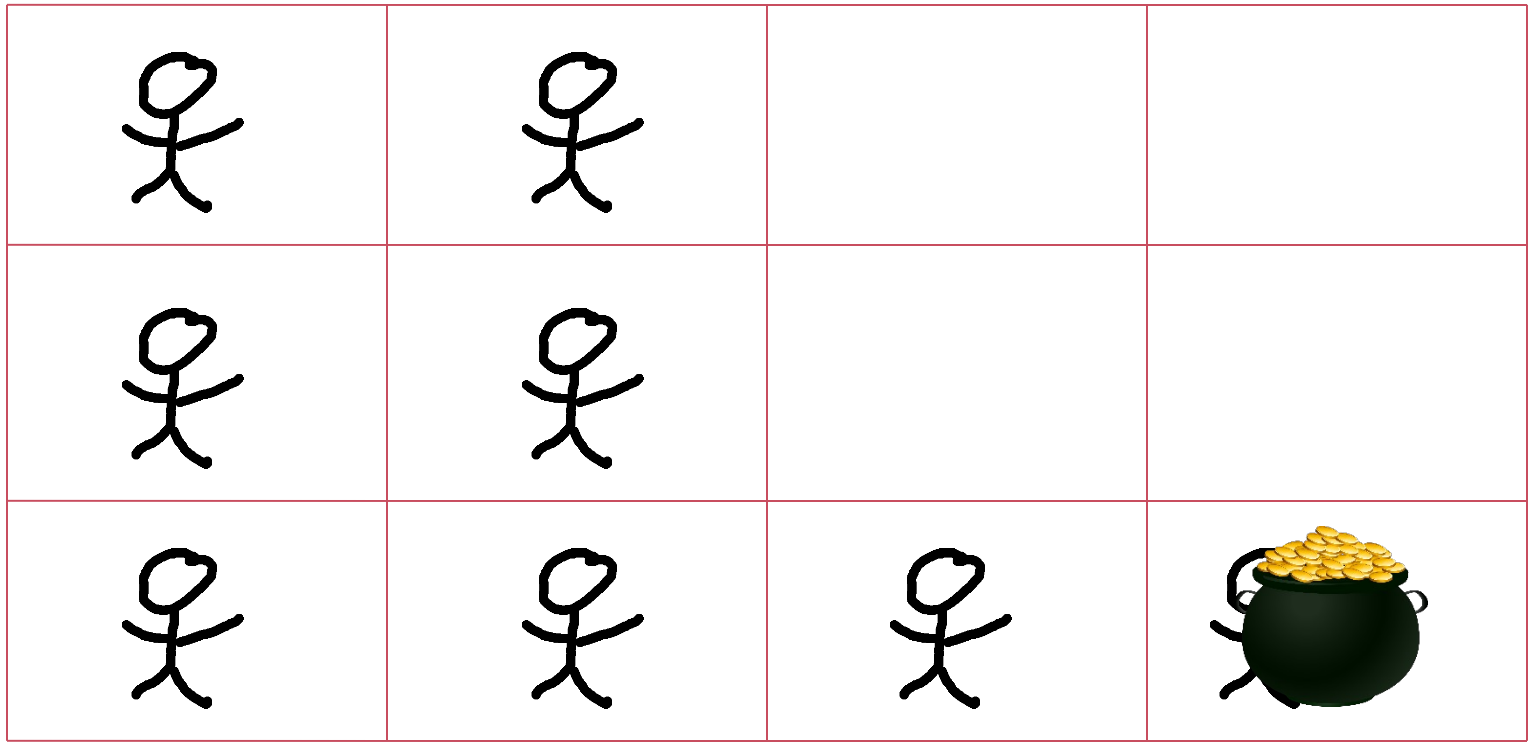
- Reinforcement learning (RL) was studied in animals by Thorndike [1898].
- Became the study of Behaviorism [Skinner, 1953] (see Skinner box on the right).
- Formulated into artificial intelligence; see leading book by Sutton and Barto, 1998.
  (new draft available online http://incompleteideas.net/book/bookdraft2018jan1.pdf)



A "Skinner box"

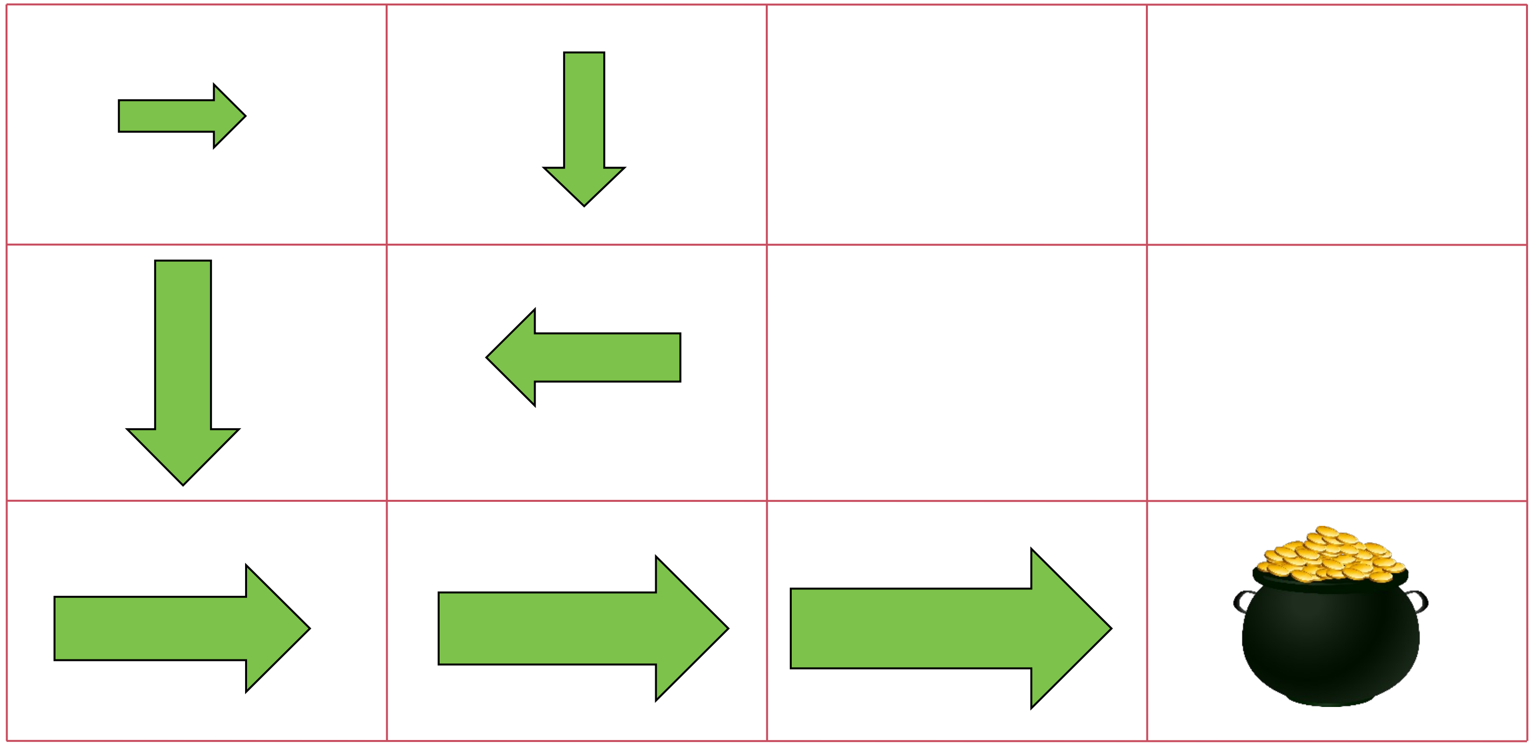
DeUmbra

Beginning with random exploration



In reinforcement learning, the agent begins by randomly exploring until it reaches its goal.

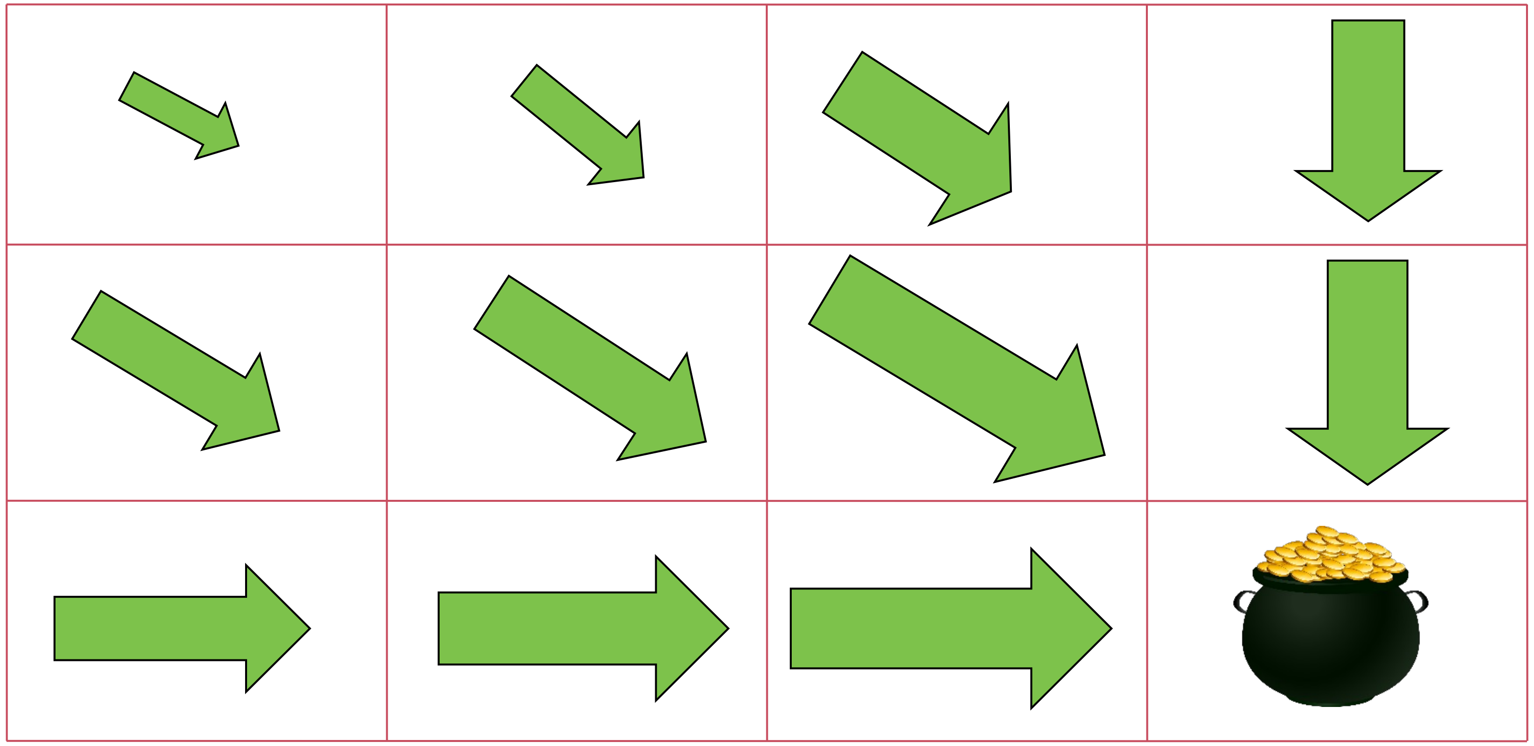
Reaching the goal



- When it reaches the goal, credit is propagated back to its previous states.
- The agent learns the function $Q^{\pi}(s, a)$, which gives the cumulative expected discounted reward of being in state s and taking action a and acting according to policy π thereafter.

DeUmbra

Learning the behavior



Eventually, the agent learns the value of being in each state and taking each action and can therefore always do the best thing in each state.

DeUmbra

RL is now like supervised learning over actions

With the rise of deep learning, we can now more effectively work in high dimensional and continuous domains, like we saw with the encoder.

In these kinds of domains, the kind of reinforcement learning that works best is policy gradient methods.

- D. Silver http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/pg.pdf
- A. Karpathy http://karpathy.github.io/2016/05/31/rl/

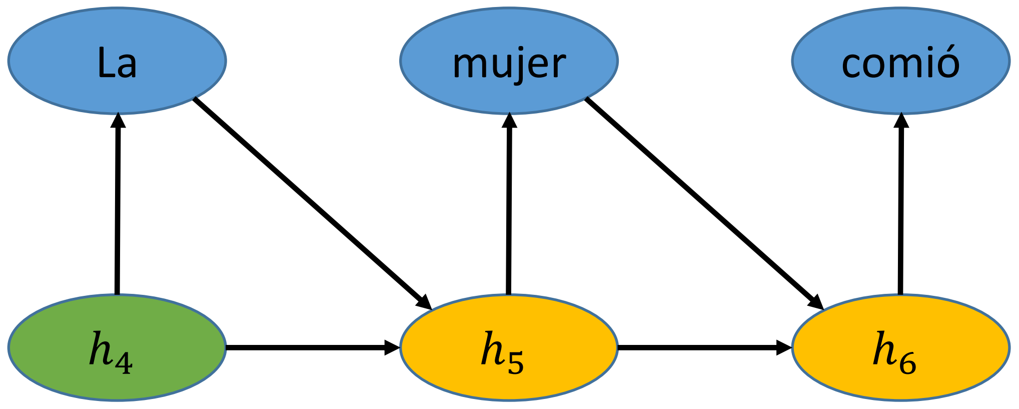
You learn a policy directly on top of a neural network. Basically softmax on top of a neural network.

So reinforcement learning starts to look more like supervised learning, where the labels are actions.

Instead of doing gradient descent on the error function as in supervised learning, you are doing gradient ascent on the expected reward.

Searching forward to get this reward

You have to do a search forward to get the state.



Since we are not doing teacher forcing where we know what the next word should be, we do a forward search: If we were to write "comió" and followed our current policy after that, how good would the final reward be?

That "how good" is the estimate we use to determine whether writing "comió" should be the action this network takes in the future.
[see Lantao et al., https://arxiv.org/pdf/1609.05473v3.pdf]

Called Monte Carlo search

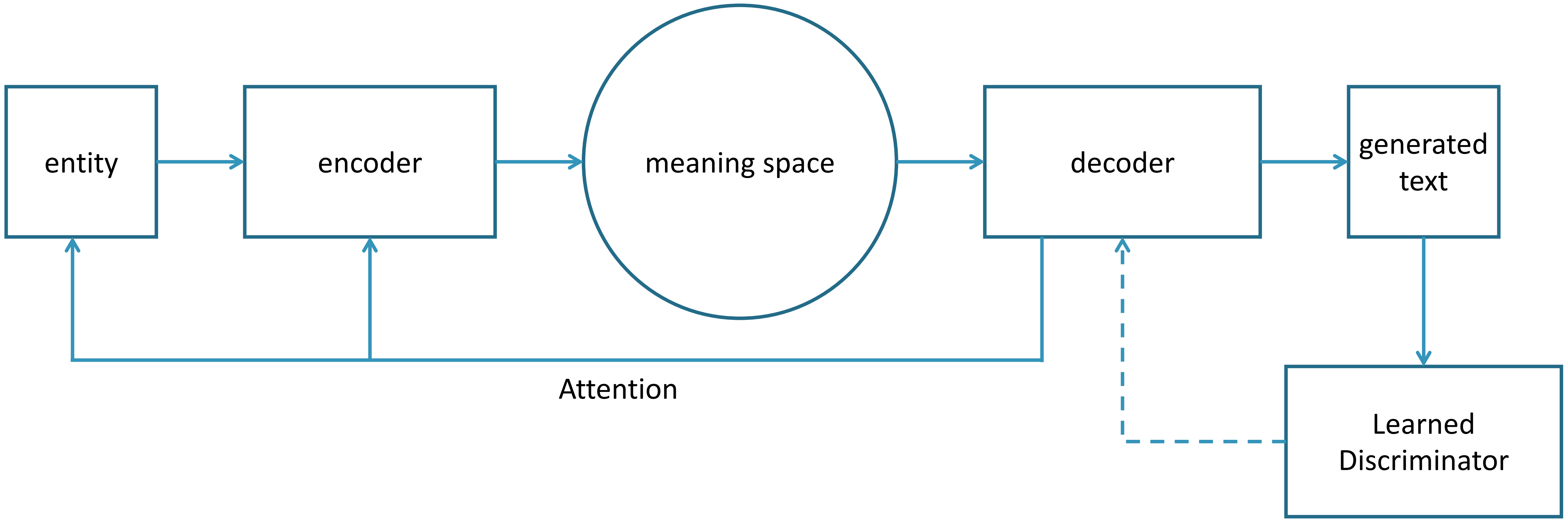
This is similar how game playing AI is done, for example with the recent success in Go
[see Silver et al., https://arxiv.org/pdf/1712.01815.pdf]

In the language domain, this kind of training can be very slow.

Outline

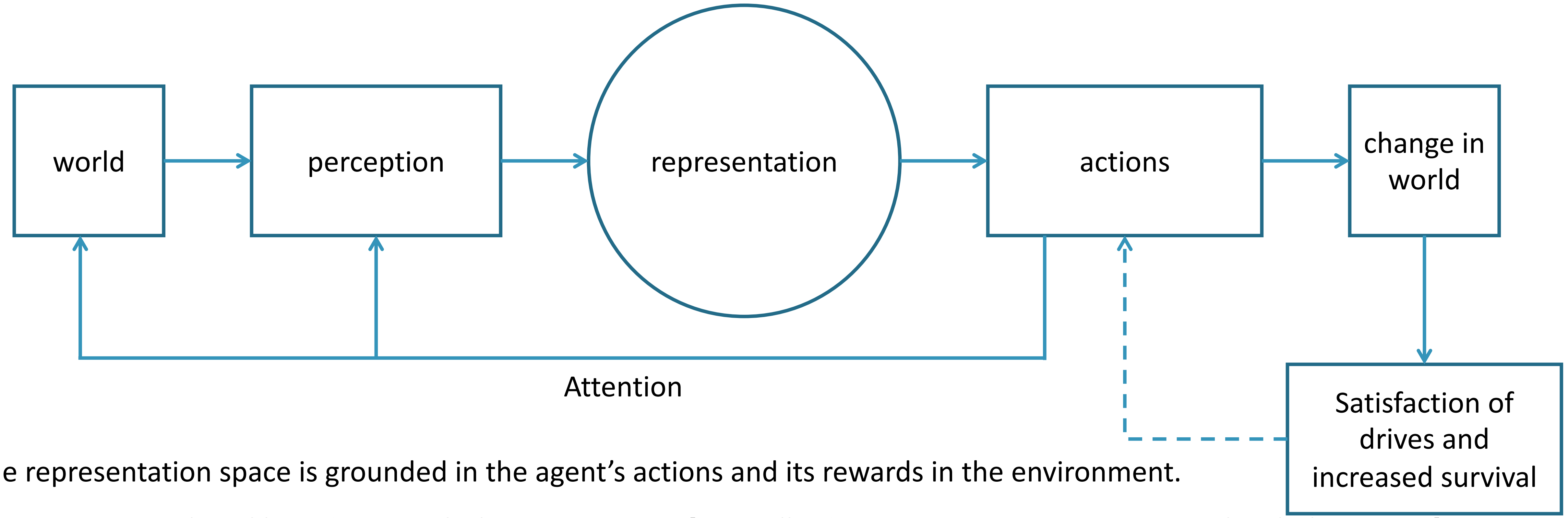
- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

Abstracting to generative models of behavior



DeUmbra

Abstracting to generative models of behavior



The representation space is grounded in the agent's actions and its rewards in the environment.

Attention can make it like a dance with the environment [see Ballard, *Brain Computation as Hierarchical Abstraction*]

But this is too simple

- Need multiple paths for perception (E.g., paths for "what" vs. "where" in visual system)
- Need hierarchies of actions

Important: need long term planning.

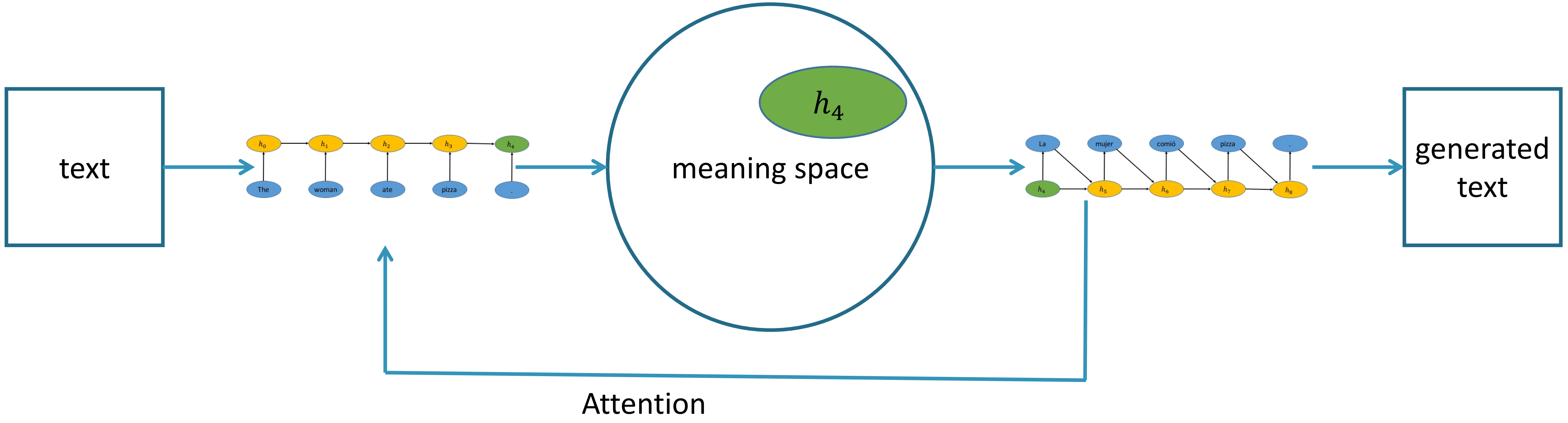
- To bring it back to neural text generation, need planning of documents, outlines

Outline

- From seq2seq to encoder-decoder
- Encoders: RNNs and CNNs
- Meaning space: generative models through variational methods
- Decoders: teacher forcing, GANs, and reinforcement learning
- A general model of behavior
- Conclusion: a sequence of abstractions

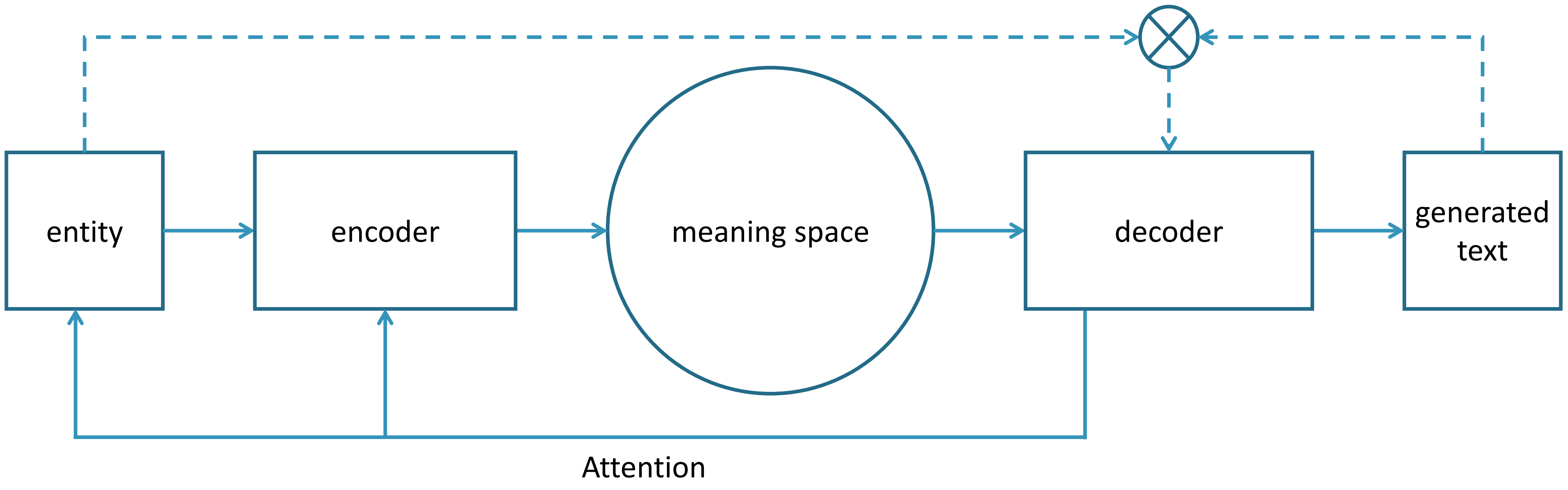
DeUmbra

Conclusion



We have shown how a system can take input from the environment, learn how to organize it into a representation, and then learn to act based on that representation.

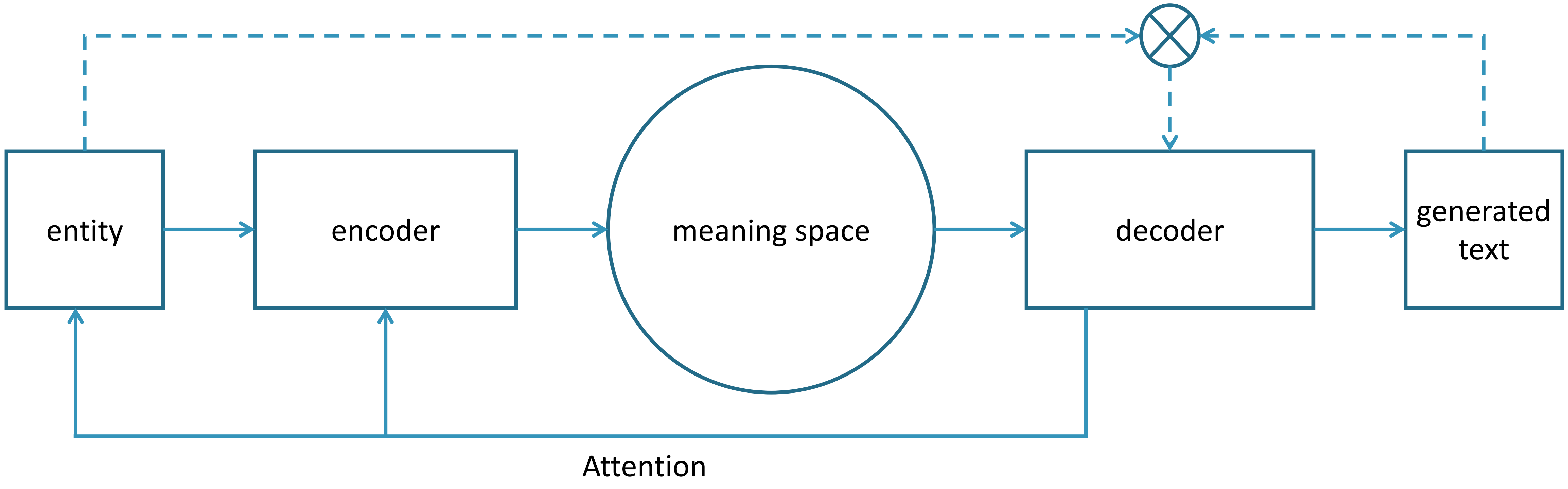
Conclusion



We have shown how a system can take input from the environment, learn how to organize it into a representation, and then learn to act based on that representation.

DeUmbra

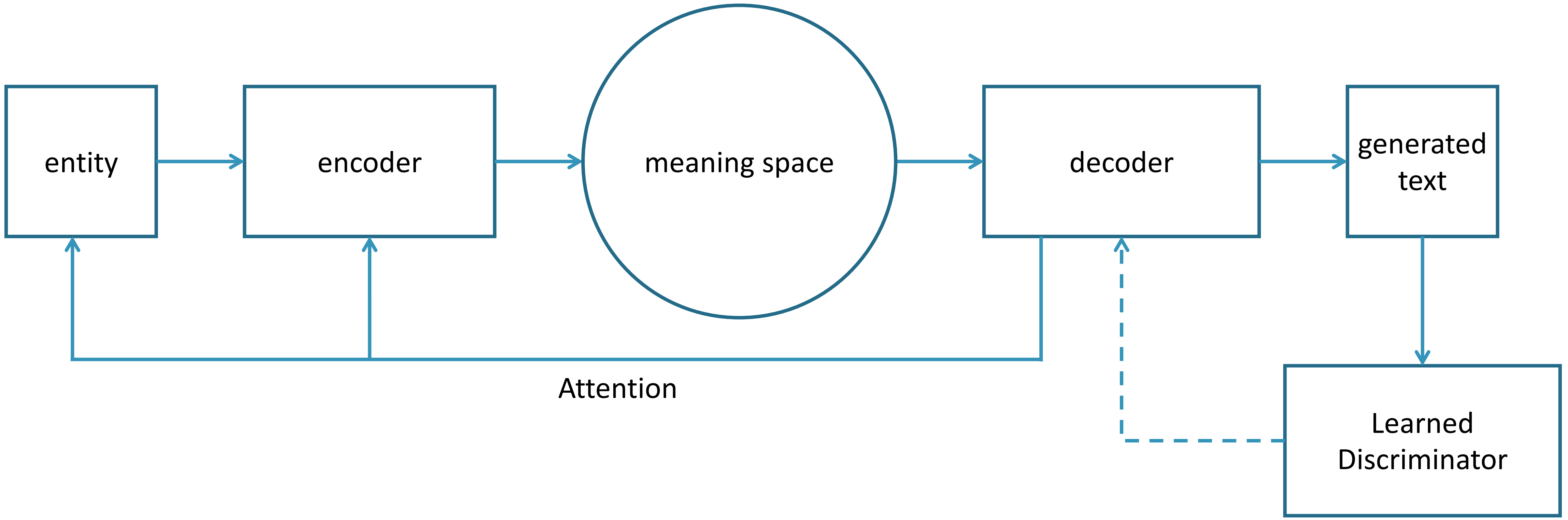
Conclusion



This process can either be learned through pairs of inputs and outputs ...

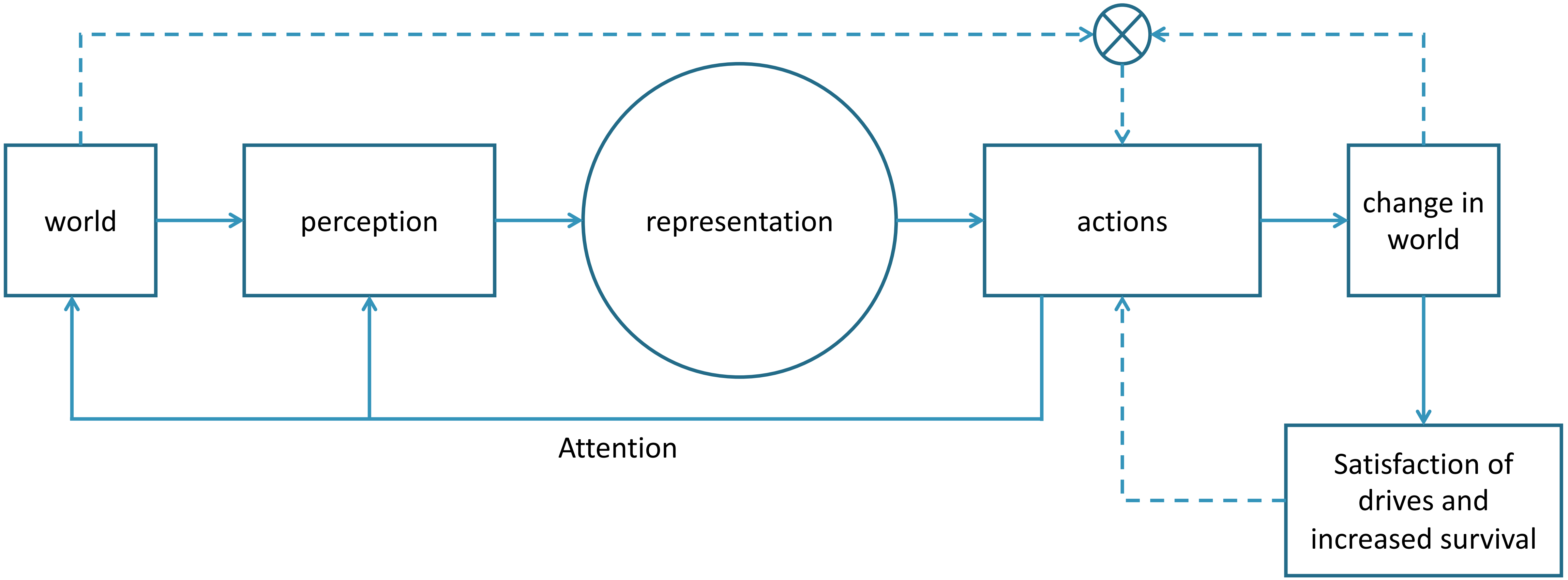
DeUmbra

Conclusion



... or by a scoring function that specifies how good a behavior was or a particular input.

Conclusion



And this process seems to generalize beyond text generation to broader actions.

Thanks for listening!
@jmugan

DeUmbra

6500 River Place Blvd.
Building 3, Suite 120
Austin, TX 78730
www.DeUmbra.com