ELEC-E5510 Speech Recognition

Recurrent Neural Networks in Language Modeling

Tamás Grósz

Department of Signal Processing and Acoustics

Introduction

Why not just n-grams?

The french boy lived in the capital city, ?

Why not just n-grams?

The french boy lived in the capital city, Paris

Why not just n-grams?

The french boy lived in the capital city, Paris

Why not just n-grams?

The french boy lived in the capital city, Paris

- We would need at least 8-grams to cover this context

Why not just n-grams?

The french boy lived in the capital city, Paris

- We would need at least 8-grams to cover this context
- High order n-grams are rare

Why not just n-grams?

The french boy lived in the capital city, Paris

- We would need at least 8-grams to cover this context
- High order n-grams are rare
- Using large n-grams is inefficient

Why not just n-grams?

The french boy lived in the capital city, Paris

- We would need at least 8-grams to cover this context
- High order n-grams are rare
- Using large n-grams is inefficient
- Google's English n-grams: 24 GB, up to 5-grams (that appear at least 40 times)

Neural Language Models

- Neural Networks could be used for LM

Neural Language Models

- Neural Networks could be used for LM
- The issues that we need to solve:

Neural Language Models

- Neural Networks could be used for LM
- The issues that we need to solve:
    1. How to input words to the network

Neural Language Models

- Neural Networks could be used for LM
- The issues that we need to solve:
    1. How to input words to the network
    2. How to remember long sequences/ distant words?

Neural Language Models

- Neural Networks could be used for LM
- The issues that we need to solve:
    1. How to input words to the network
    2. How to remember long sequences/ distant words?
    3. How to train is efficiently?

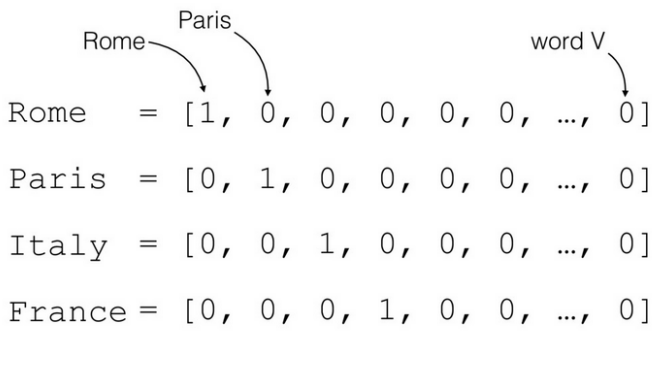
Neural LM, input words

One-hot Encoding

Idea: Let's convert the word-id to a one-hot vector!

One-hot Encoding

Idea: Let's convert the word-id to a one-hot vector!



Picture by Marco Bonzanini

Exercise 1.

We have a simple neural network, which knows 100K words, and contains 2 hidden layers with 100 neurons. This network uses the one-hot embedding to process words.

Exercise 1.

We have a simple neural network, which knows 100K words, and contains 2 hidden layers with 100 neurons. This network uses the one-hot embedding to process words.

- Question 1. What percentage of the parameters are in the input and output layers?

Exercise 1.

We have a simple neural network, which knows 100K words, and contains 2 hidden layers with 100 neurons. This network uses the one-hot embedding to process words.

- Question 1. What percentage of the parameters are in the input and output layers?
- Question 2. If we want to cover 10 past words with this network how would the percentage change?

Exercise 1.

We have a simple neural network, which knows 100K words, and contains 2 hidden layers with 100 neurons. This network uses the one-hot embedding to process words.

- Question 1. What percentage of the parameters are in the input and output layers?
- Question 2. If we want to cover 10 past words with this network how would the percentage change?
- Question 3. How would you change this model to reduce its size and increase its speed?

Exercise 1.

We have a simple neural network, which knows 100K words, and contains 2 hidden layers with 100 neurons. This network uses the one-hot embedding to process words.

- Question 1. What percentage of the parameters are in the input and output layers?
- Question 2. If we want to cover 10 past words with this network how would the percentage change?
- Question 3. How would you change this model to reduce its size and increase its speed?
- Don't forget to submit the answers in MyCourse!

Issues with the one-hot embedding

- If we have 10M words, and 1000 hidden neurons, what is the #parameters=?

Issues with the one-hot embedding

- If we have 10M words, and 1000 hidden neurons, what is the #parameters=? (10M*1000 = 10B)
- The one-hot vectors have no relation to each other (everything is equally different)

Issues with the one-hot embedding

- If we have 10M words, and 1000 hidden neurons, what is the #parameters=? (10M*1000 = 10B)
- The one-hot vectors have no relation to each other (everything is equally different)
- Solution: word2vec

Issues with the one-hot embedding

- If we have 10M words, and 1000 hidden neurons, what is the #parameters=? (10M*1000 = 10B)
- The one-hot vectors have no relation to each other (everything is equally different)
- Solution: word2vec
  - Basic idea: create a small continuous vector representations

Issues with the one-hot embedding

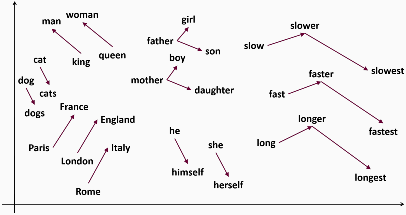
- If we have 10M words, and 1000 hidden neurons, what is the #parameters=? (10M*1000 = 10B)
- The one-hot vectors have no relation to each other (everything is equally different)
- Solution: word2vec
    - Basic idea: create a small continuous vector representations
    - Autoencoder-based solutions (word in, same word out)

Issues with the one-hot embedding

- If we have 10M words, and 1000 hidden neurons, what is the #parameters=? (10M*1000 = 10B)
- The one-hot vectors have no relation to each other (everything is equally different)
- Solution: word2vec
  - Basic idea: create a small continuous vector representations
  - Autoencoder-based solutions (word in, same word out)
  - Similar words will have similar representation

Issues with the one-hot embedding

- If we have 10M words, and 1000 hidden neurons, what is the #parameters=? (10M*1000 = 10B)
- The one-hot vectors have no relation to each other (everything is equally different)
- Solution: word2vec
  - Basic idea: create a small continuous vector representations
  - Autoencoder-based solutions (word in, same word out)
  - Similar words will have similar representation



Picture by Samy Zafrany

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.
I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.
I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

There are several ways of getting the sub-word units:

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.
I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

There are several ways of getting the sub-word units:

- Byte pair encoding (BPE)

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.

I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

There are several ways of getting the sub-word units:

- Byte pair encoding (BPE)
- Morfessor

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.
I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

There are several ways of getting the sub-word units:

- Byte pair encoding (BPE)
- Morfessor
- Sentence/word-piece

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.
I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

There are several ways of getting the sub-word units:

- Byte pair encoding (BPE)
- Morfessor
- Sentence/word-piece
- All uses the basic idea of building a subword vocabulary that covers some training text well

Sub-words

To reduce the size of the input layer, one can switch to using sub-words or characters.
I saw a girl with a telescope : I saw a girl with a te+le+s+c+o+pe

There are several ways of getting the sub-word units:

- Byte pair encoding (BPE)
- Morfessor
- Sentence/word-piece
- All uses the basic idea of building a subword vocabulary that covers some training text well
- Using a few thousand units could cover a large vocabulary

Neural LM, long context

Using context with NNLM

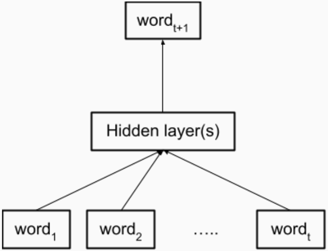
Naive solution:

Using context with NNLM

Naive solution: just connect all words to the first hidden layer.

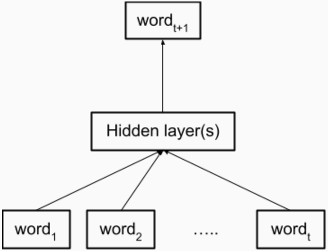
Using context with NNLM

Naive solution: just connect all words to the first hidden layer.



Using context with NNLM

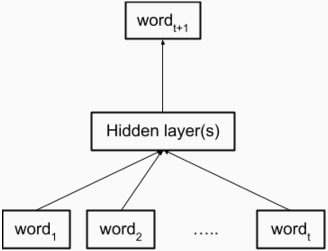
Naive solution: just connect all words to the first hidden layer.



- Too many parameters: #words*#neurons*context

Using context with NNLM

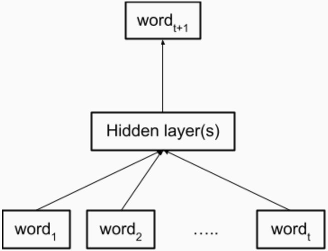
Naive solution: just connect all words to the first hidden layer.



- Too many parameters: #words*#neurons*context
- Increasing the context grows the network!

Using context with NNLM

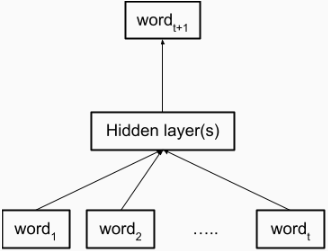
Naive solution: just connect all words to the first hidden layer.



- Too many parameters: #words*#neurons*context
- Increasing the context grows the network!
- We lose the temporal info (not time-shift invariant)

Using context with NNLM

Naive solution: just connect all words to the first hidden layer.



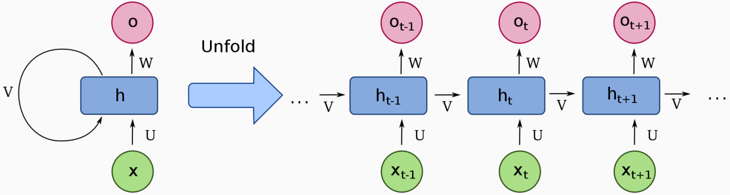
- Too many parameters: #words*#neurons*context
- Increasing the context grows the network!
- We lose the temporal info (not time-shift invariant)

Recurrent neurons

Recurrent neurons: new type of neurons to handle time series through a "recurrent" connection.

Recurrent neurons

Recurrent neurons: new type of neurons to handle time series through a "recurrent" connection.



Picture by Wikipedia

LSTM cell

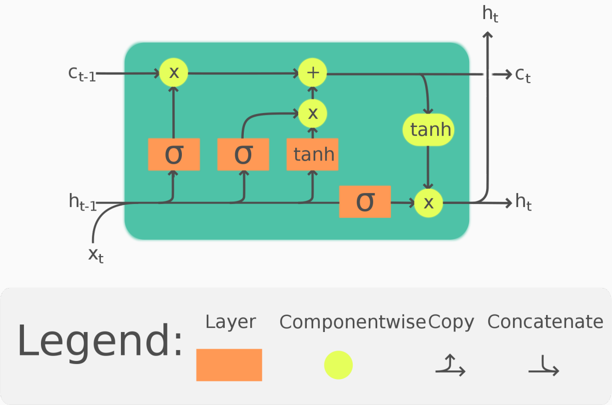
Having a recurrent connection is not enough, we need long-term memory!

LSTM cell

Having a recurrent connection is not enough, we need long-term memory! RNNs are vulnerable to the "vanishing gradient". (the long-term gradients could get close to 0, or explode)

LSTM cell

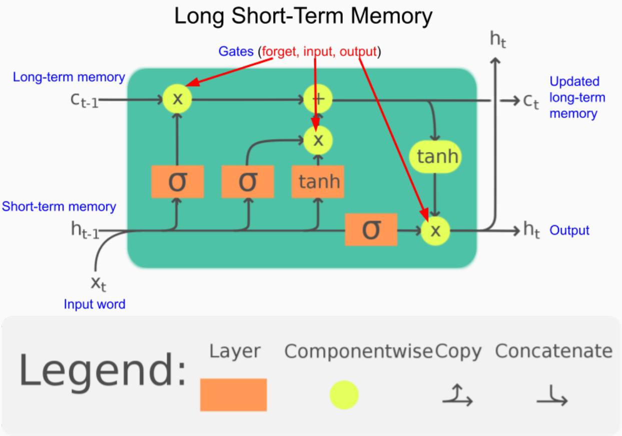
Having a recurrent connection is not enough, we need long-term memory! RNNs are vulnerable to the "vanishing gradient". (the long-term gradients could get close to 0, or explode)



Picture by Wikipedia

LSTM cell

Having a recurrent connection is not enough, we need long-term memory! RNNs are vulnerable to the "vanishing gradient". (the long-term gradients could get close to 0, or explode)



Picture by Wikipedia

Attention

- Recurrent models are slow and hard to train.

Attention

- Recurrent models are slow and hard to train.
- In LSTMs the long-term memory could be forgotten or overwritten.

Attention

- Recurrent models are slow and hard to train.
- In LSTMs the long-term memory could be forgotten or overwritten.
- Alternative solution: use feed-forward models with attention.

Attention

- Recurrent models are slow and hard to train.
- In LSTMs the long-term memory could be forgotten or overwritten.
- Alternative solution: use feed-forward models with attention.

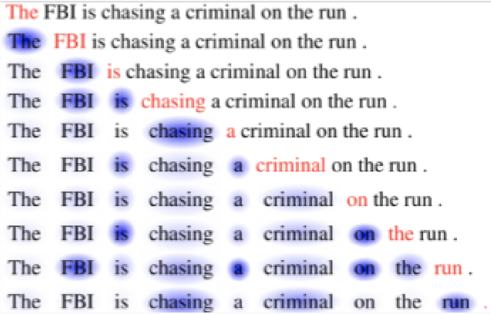
Attention mechanism

The core idea is that the model should have access to all inputs instead of just the last one and learn to "pay attention" to the relevant parts/words.

- Recurrent models are slow and hard to train.
- In LSTMs the long-term memory could be forgotten or overwritten.
- Alternative solution: use feed-forward models with attention.

Attention mechanism

The core idea is that the model should have access to all inputs instead of just the last one and learn to "pay attention" to the relevant parts/words.



Attention

How can we calculate the attention values?

Attention

How can we calculate the attention values?

- We need 3 component: **Q**uery, **K**ey, **V**alue

Attention

How can we calculate the attention values?

- We need 3 component: **Q**uery, **K**ey, **V**alue
- Query: the embedding of the last word

Attention

How can we calculate the attention values?

- We need 3 component: **Q**uery, **K**ey, **V**alue
- Query: the embedding of the last word
- Key: the embedding of other words

Attention

How can we calculate the attention values?

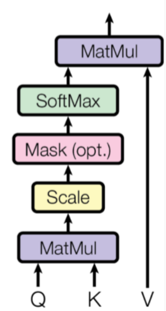
- We need 3 component: **Q**uery, **K**ey, **V**alue
- Query: the embedding of the last word
- Key: the embedding of other words
- Value: additional transformation of the words.

Attention

How can we calculate the attention values?

- We need 3 component: **Q**uery, **K**ey, **V**alue
- Query: the embedding of the last word
- Key: the embedding of other words
- Value: additional transformation of the words.
- We use Q and K to get the attention values:

$$Attention(Q, K, V) = \underbrace{softmax}_{\text{sum is 1}} \left(\frac{\overbrace{QK^T}^{\text{Dot-product}}}{\underbrace{\sqrt{d_k}}_{\text{dimension of Q and K}}} \right) V$$



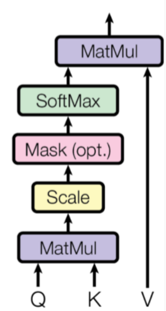
Picture from Attention is all you need

Attention

How can we calculate the attention values?

- We need 3 component: **Q**uery, **K**ey, **V**alue
- Query: the embedding of the last word
- Key: the embedding of other words
- Value: additional transformation of the words.
- We use Q and K to get the attention values:

$$Attention(Q, K, V) = \underbrace{softmax}_{\text{sum is 1}} \left(\frac{\overbrace{QK^T}^{\text{Dot-product}}}{\underbrace{\sqrt{d_k}}_{\text{dimension of Q and K}}} \right) V$$



Picture from Attention is all you need

Note: this it the dot-product attention variant. There are several other ways to compute the scores (for more see Attention-Tutorial)

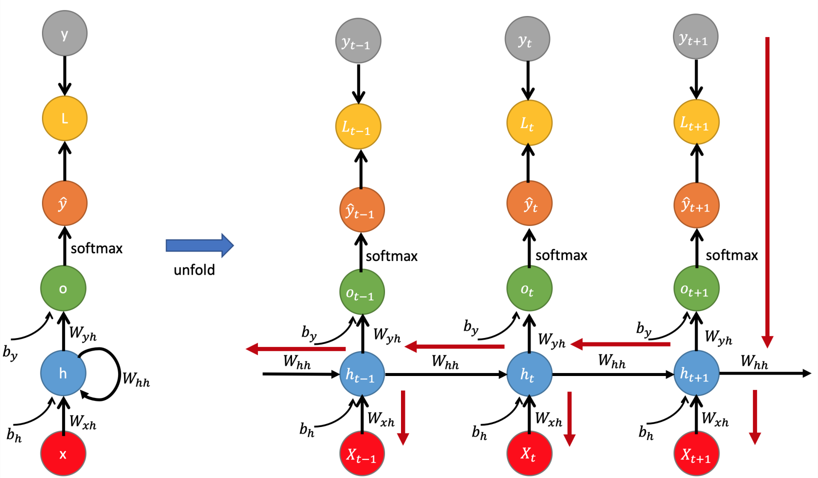
Efficient NNLM Training

Training RNNLMs

Backpropagation through time (BPTT) is a gradient-based training algorithm for RNNs.

Training RNNLMs

Backpropagation through time (BPTT) is a gradient-based training algorithm for RNNs.



Picture by Mustafa Murat Arat

Issues with training an NNLM

- Training NNLMs is a slow process.

Issues with training an NNLM

- Training NNLMs is a slow process.
- One huge issue is the softmax activation.

Issues with training an NNLM

- Training NNLMs is a slow process.
- One huge issue is the softmax activation.
- The expected/correct output is quite sparse (only one correct word)

Issues with training an NNLM

- Training NNLMs is a slow process.
- One huge issue is the softmax activation.
- The expected/correct output is quite sparse (only one correct word)
- We can exploit this to reduce the required number of computations!

Solutions

Negative sampling

Solutions

Negative sampling

1. Select a few non-target words (negative samples).

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.
3. Update only these output units.

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.
3. Update only these output units.

Noise Contrastive Estimation

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.
3. Update only these output units.

Noise Contrastive Estimation

1. Very similar to negative sampling.

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.
3. Update only these output units.

Noise Contrastive Estimation

1. Very similar to negative sampling.
2. NCE uses a Logistic Regression to determine which words are real and which are noise (negative sample).

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.
3. Update only these output units.

Noise Contrastive Estimation

1. Very similar to negative sampling.
2. NCE uses a Logistic Regression to determine which words are real and which are noise (negative sample).
3. Main differences:
   - Sigmoid transformation instead of softmax

Solutions

Negative sampling

1. Select a few non-target words (negative samples).
2. Pretend that the target word and the negative samples represent the entire vocabulary.
3. Update only these output units.

Noise Contrastive Estimation

1. Very similar to negative sampling.
2. NCE uses a Logistic Regression to determine which words are real and which are noise (negative sample).
3. Main differences:
   - Sigmoid transformation instead of softmax
   - **Binary** Cross Entropy Loss

Using NNLM in ASR

How to use NNLMs?

How can we use the NNLMs in an ASR system?

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
    - Possible, but complicated (the search space could explode)

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
    - Possible, but complicated (the search space could explode)
    - Requires special decoders and a lot of work

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
    - Possible, but complicated (the search space could explode)
    - Requires special decoders and a lot of work
2. N-best re-scoring

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
   - Possible, but complicated (the search space could explode)
   - Requires special decoders and a lot of work
2. N-best re-scoring
   - Easiest option, after decoding with an n-gram generate the **n** most probable texts

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
   - Possible, but complicated (the search space could explode)
   - Requires special decoders and a lot of work
2. N-best re-scoring
   - Easiest option, after decoding with an n-gram generate the **n** most probable texts
   - Score n-best alternatives with NNLM to get the most probable one

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
    - Possible, but complicated (the search space could explode)
    - Requires special decoders and a lot of work
2. N-best re-scoring
    - Easiest option, after decoding with an n-gram generate the **n** most probable texts
    - Score n-best alternatives with NNLM to get the most probable one
3. Lattice re-scoring

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
    - Possible, but complicated (the search space could explode)
    - Requires special decoders and a lot of work
2. N-best re-scoring
    - Easiest option, after decoding with an n-gram generate the **n** most probable texts
    - Score n-best alternatives with NNLM to get the most probable one
3. Lattice re-scoring
    - Generate a decoded lattice with n-gram

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
   - Possible, but complicated (the search space could explode)
   - Requires special decoders and a lot of work
2. N-best re-scoring
   - Easiest option, after decoding with an n-gram generate the **n** most probable texts
   - Score n-best alternatives with NNLM to get the most probable one
3. Lattice re-scoring
   - Generate a decoded lattice with n-gram
   - Replace the LM probabilities with NNLM estimates

How to use NNLMs?

How can we use the NNLMs in an ASR system?

1. By replacing the n-gram model
   - Possible, but complicated (the search space could explode)
   - Requires special decoders and a lot of work
2. N-best re-scoring
   - Easiest option, after decoding with an n-gram generate the **n** most probable texts
   - Score n-best alternatives with NNLM to get the most probable one
3. Lattice re-scoring
   - Generate a decoded lattice with n-gram
   - Replace the LM probabilities with NNLM estimates
   - Could be slow if the lattice is large

How to use NNLMs?

How can we use the NNLMs in an ASR system?

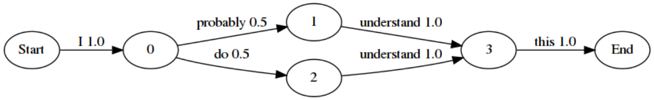
1. By replacing the n-gram model
   - Possible, but complicated (the search space could explode)
   - Requires special decoders and a lot of work
2. N-best re-scoring
   - Easiest option, after decoding with an n-gram generate the **n** most probable texts
   - Score n-best alternatives with NNLM to get the most probable one
3. Lattice re-scoring
   - Generate a decoded lattice with n-gram
   - Replace the LM probabilities with NNLM estimates
   - Could be slow if the lattice is large

What is a lattice?

A lattice could be quite simple:

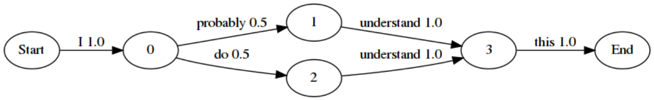
What is a lattice?

A lattice could be quite simple:



What is a lattice?

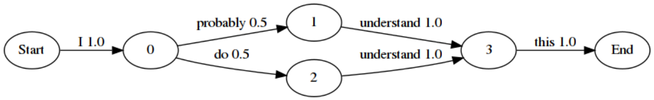
A lattice could be quite simple:



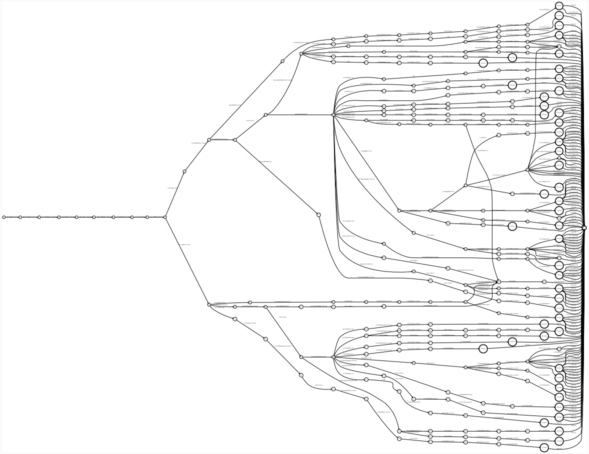
But reality is often ugly:

What is a lattice?

A lattice could be quite simple:



But reality is often ugly:



Summary

The main topics briefly explained in this presentation:

1. NNLM
2. Recurrent models
3. Attention
4. Techniques to make the training efficient