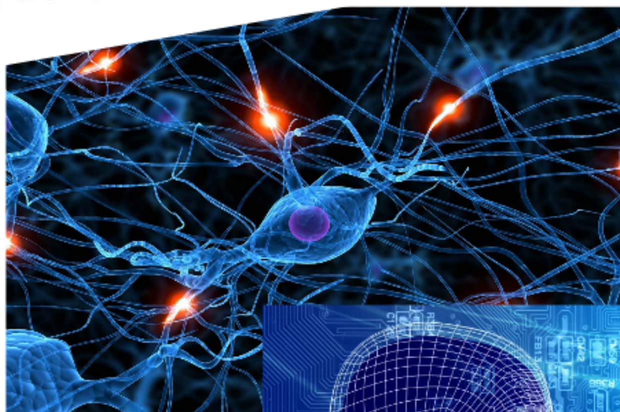
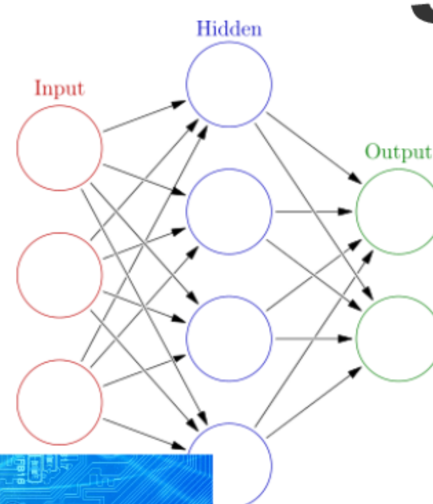
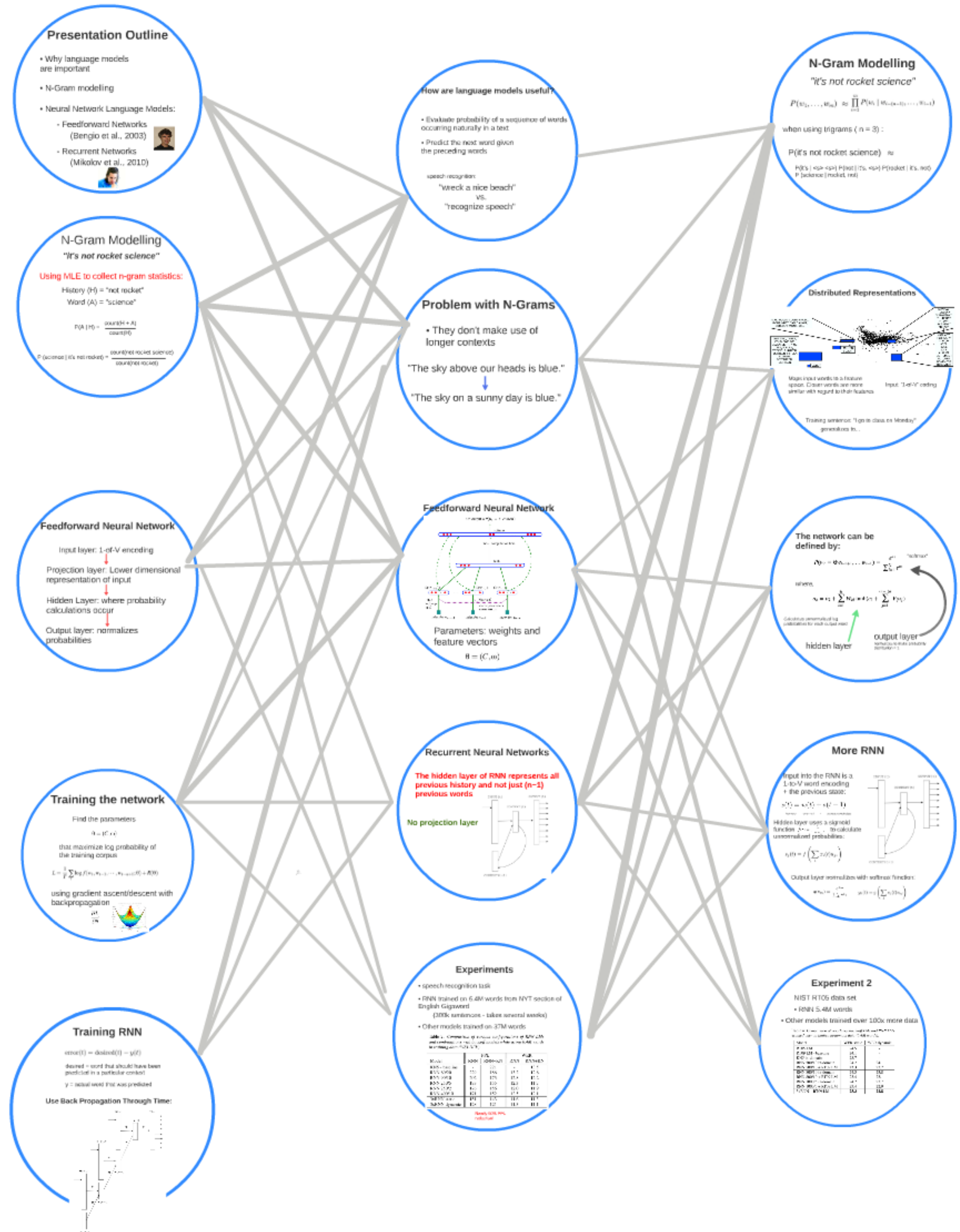
Neural Network Language Modelling

$$\prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$







Presentation Outline
• Why language models are important
• N-Gram modeling
• Neural Network Language Models:
- Feedforward Networks (Bengio et al., 2003)
- Recurrent Networks (Mikolov et al., 2010)
N-Gram Modelling
"it's not rocket science"
Using MLE to collect n-gram statistics:
History (H) = "not rocket"
Word (A) = "science"
Feedforward Neural Network
Input layer: 1-of-V encoding
Projection layer: Lower dimensional representation of input
Hidden Layer: where probability calculations occur
Output layer: normalizes probabilities
Training the network
Find the parameters
that maximize log probability of the training corpus
using gradient ascent/descent with backpropagation
Training RNN
Use Back Propagation Through Time:
How are language models useful?
• Evaluate probability of a sequence of words occurring naturally in a text
• Predict the next word given the preceding words
speech recognition
"wreck a nice beach"
vs.
"recognize speech"
Problem with N-Grams
• They don't make use of longer contexts
"The sky above our heads is blue."
"The sky on a sunny day is blue."
Feedforward Neural Network
Parameters: weights and feature vectors
Recurrent Neural Networks
The hidden layer of RNN represents all previous history and not just (n−1) previous words
No projection layer
Experiments
• speech recognition task
• RNN trained on 6.4M words from NYT section of English Gigaword
(300k sentences - taken several weeks)
• Other models trained on 37M words
N-Gram Modelling
"it's not rocket science"
when using trigrams (n = 3) :
P(it's not rocket science) ≈
Distributed Representations
Input: "1-of-V" coding
The network can be defined by:
where,
hidden layer
output layer
More RNN
Input into the RNN is a 1-to-V word encoding + the previous state:
Output layer normalizes with softmax function
Experiment 2
NIST RT05 data set
• RNN 5.4M words
• Other models trained over 100x more data

Presentation Outline

- Why language models are important

- N-Gram modelling

- Neural Network Language Models:

  - Feedforward Networks (Bengio et al., 2003)

  - Recurrent Networks (Mikolov et al., 2010)

How are language models useful?

- Evaluate probability of a sequence of words occurring naturally in a text
- Predict the next word given the preceding words

speech recognition:

"wreck a nice beach"
vs.
"recognize speech"

N-Gram Modelling

"it's not rocket science"

$$P(w_1, \ldots, w_m) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

when using trigrams (n = 3) :

P(it's not rocket science) $\approx$

P(it's | <s> <s>) P(not | it's, <s>) P(rocket | it's, not)
P (science | rocket, not)

N-Gram Modelling

"it's not rocket science"

Using MLE to collect n-gram statistics:

History (H) = "not rocket"

Word (A) = "science"

$$P(A \mid H) = \frac{\text{count}(H + A)}{\text{count}(H)}$$

$$P\,(\text{science} \mid \text{it's not rocket}) = \frac{\text{count(not rocket science)}}{\text{count(not rocket)}}$$

Problem with N-Grams

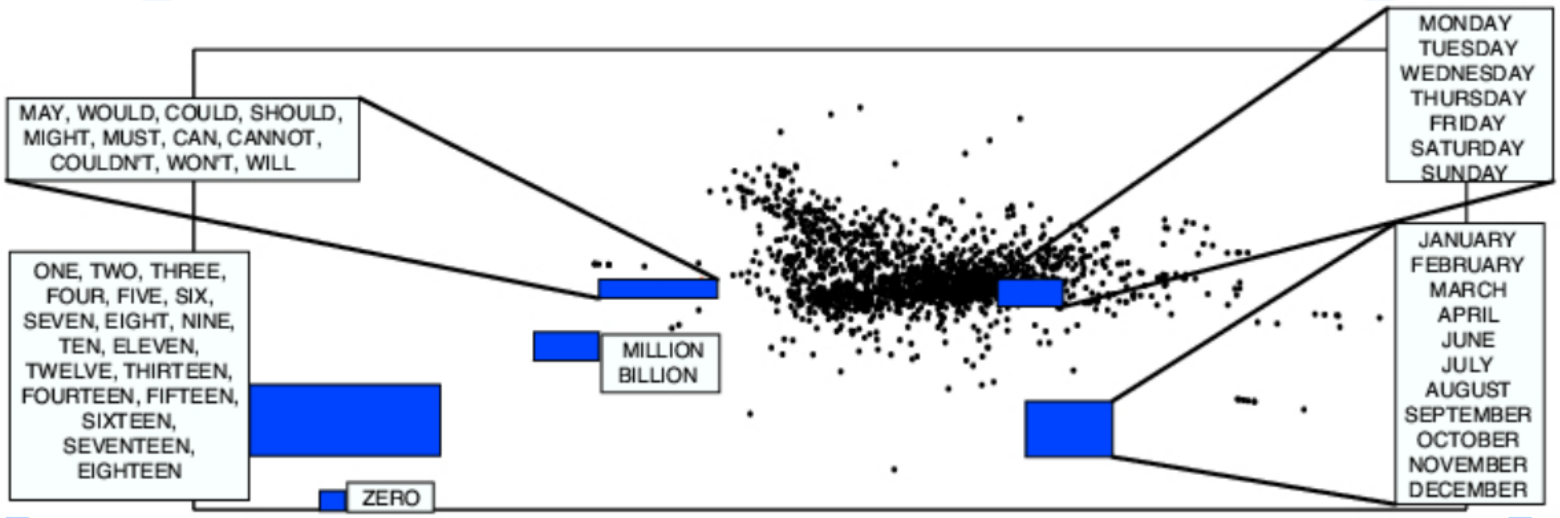
- They don't make use of longer contexts

"The sky above our heads is blue."

↓

"The sky on a sunny day is blue."

Distributed Representations



Maps input words to a feature space. Closer words are more similar with regard to their features

Input: '1-of-V' coding

Training sentence: "I go to class on Monday"

generalizes to...

Feedforward Neural Network

Input layer: 1-of-V encoding

↓

Projection layer: Lower dimensional representation of input

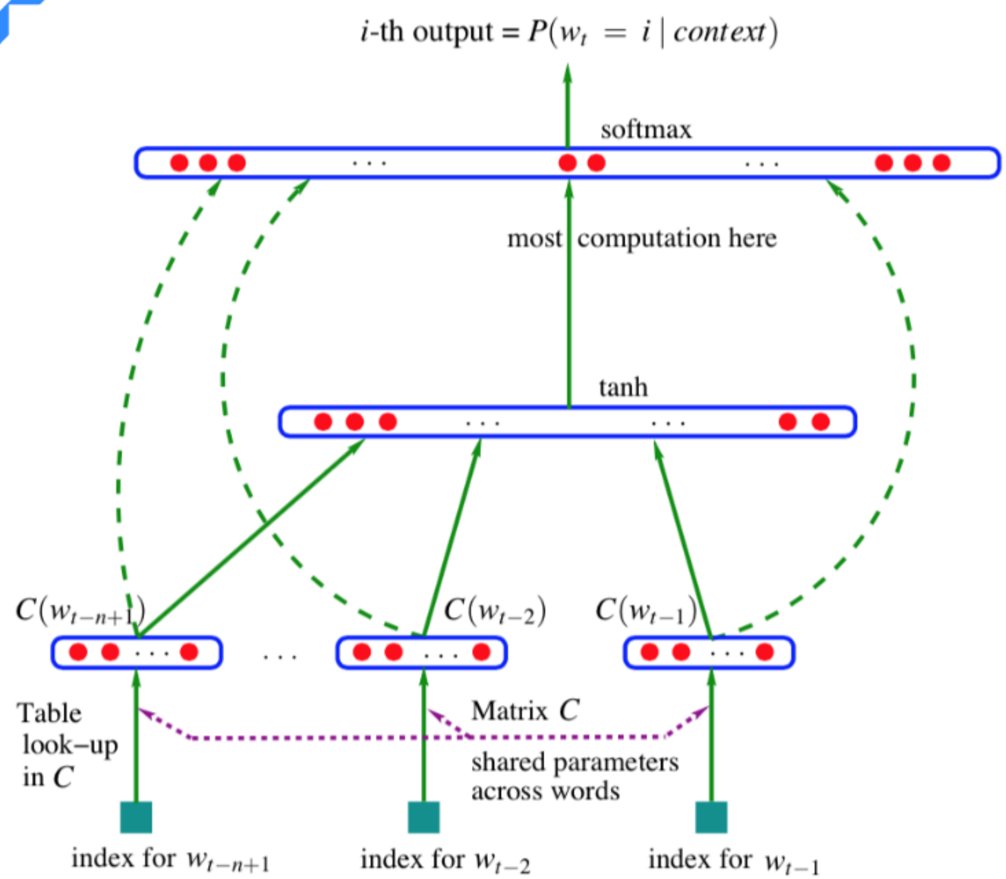
↓

Hidden Layer: where probability calculations occur

↓

Output layer: normalizes probabilities

Feedforward Neural Network



Parameters: weights and feature vectors

$$\theta = (C, \omega)$$

The network can be defined by:

$$P(w_t = k | w_{t-n+1}, \dots w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^{N} e^{a_l}}$$

"softmax"

where,

$$a_k = b_k + \sum_{i=1}^{h} W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j)$$

Calculates unnormalized log probabilities for each output word

hidden layer

output layer

normalizes to make probability distribution = 1

Training the network

Find the parameters

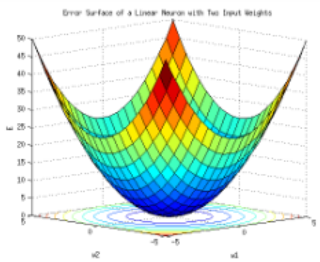
$$\theta = (C, \omega)$$

that maximize log probability of
the training corpus

$$L = \frac{1}{T}\sum_t \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}; \theta) + R(\theta)$$

using gradient ascent/descent with
backpropagation

$$\frac{\partial L}{\partial \theta}$$



Experiment

- Trained on 800,000 words in Brown corpus

-24% lower perplexity than modified Kneser-Kney

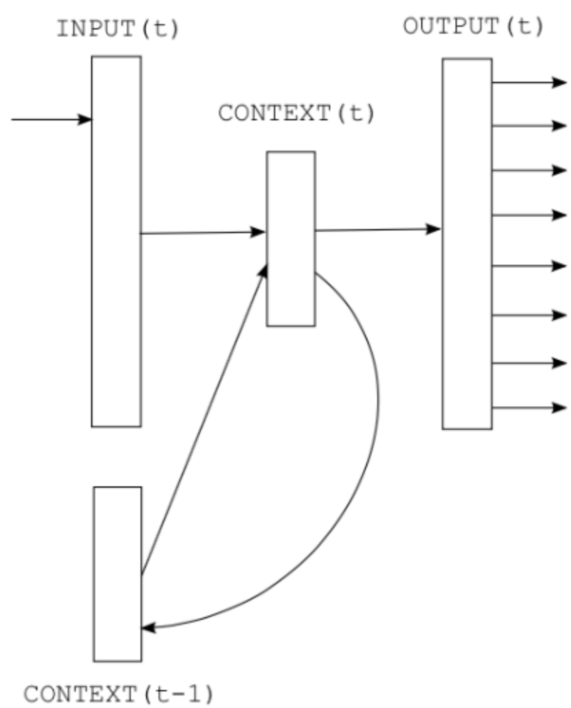
- Context length of 5 worked best for the feedforward model

- Trigrams worked best for the Kneser-Kney

Do feedforward networks really make use of longer contexts?

Recurrent Neural Networks

The hidden layer of RNN represents all previous history and not just (n−1) previous words



No projection layer

More RNN

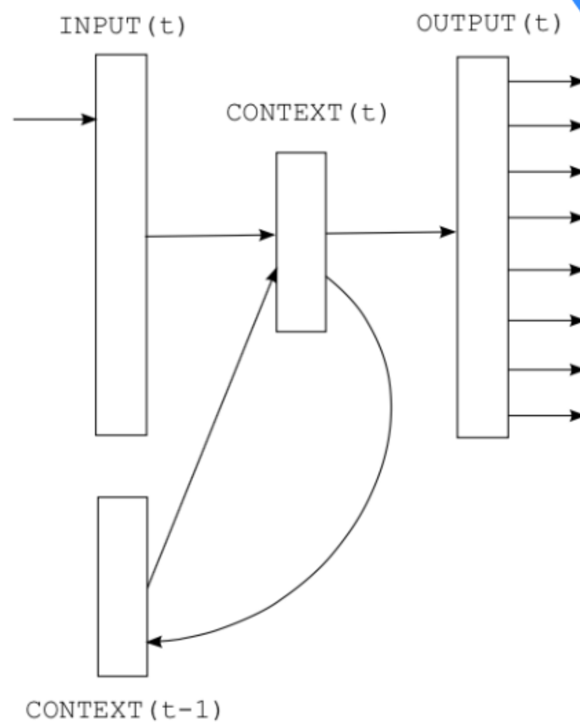
Input into the RNN is a 1-to-V word encoding + the previous state:

$$x(t) = w(t) + s(t-1)$$

input vector = current word + previous network state

Hidden layer uses a sigmoid function $f(z) = \frac{1}{1+e^{-z}}$ to calculate unnormalized probabilites:

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right)$$



Output layer normalizes with softmax function:

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \qquad y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right)$$

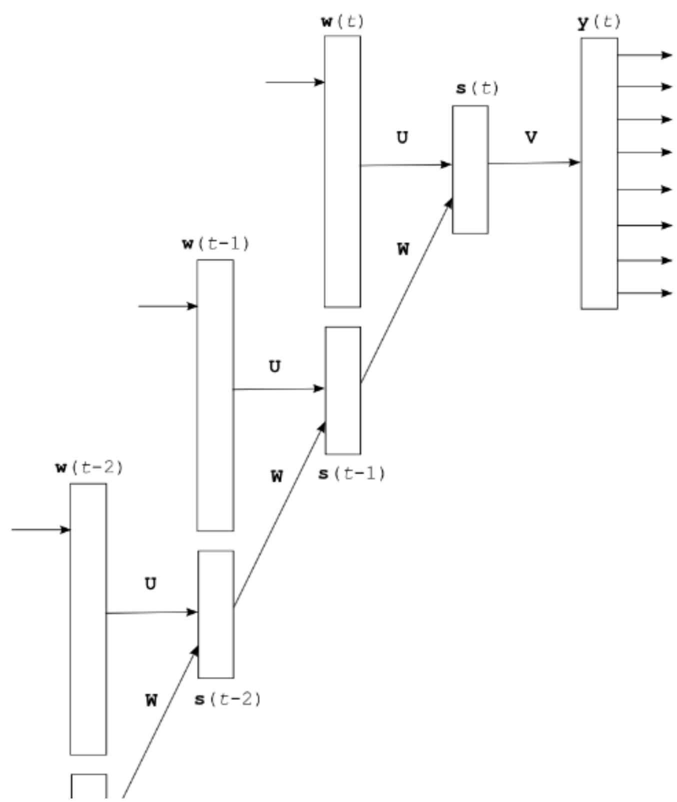
Training RNN

$$\text{error}(t) = \text{desired}(t) - y(t)$$

desired = word that should have been predicted in a particular context

y = actual word that was predicted

Use Back Propagation Through Time:



Experiments

- speech recognition task
- RNN trained on 6.4M words from NYT section of English Gigaword

  (300k sentences - takes several weeks)
- Other models trained on 37M words

Table 2: *Comparison of various configurations of RNN LMs and combinations with backoff models while using 6.4M words in training data (WSJ DEV).*

	PPL		WER	
Model	RNN	RNN+KN	RNN	RNN+KN
KN5 - baseline	-	221	-	13.5
RNN 60/20	229	186	13.2	12.6
RNN 90/10	202	173	12.8	12.2
RNN 250/5	173	155	12.3	11.7
RNN 250/2	176	156	12.0	11.9
RNN 400/10	171	152	12.5	12.1
3xRNN static	151	143	11.6	11.3
3xRNN dynamic	128	121	11.3	11.1

Nearly 50% PPL reduction!

Experiment 2

NIST RT05 data set

- RNN 5.4M words

- Other models trained over 100x more data

Table 4: *Comparison of very large back-off LMs and RNN LMs trained only on limited in-domain data (5.4M words).*

Model	WER static	WER dynamic
RT05 LM	24.5	-
RT09 LM - baseline	24.1	-
KN5 in-domain	25.7	-
RNN 500/10 in-domain	24.2	24.1
RNN 500/10 + RT09 LM	**23.3**	23.2
RNN 800/10 in-domain	24.3	23.8
RNN 800/10 + RT09 LM	23.4	23.1
RNN 1000/5 in-domain	24.2	23.7
RNN 1000/5 + RT09 LM	23.4	22.9
3xRNN + RT09 LM	**23.3**	**22.8**

questions...?