Recent advances in neural language models

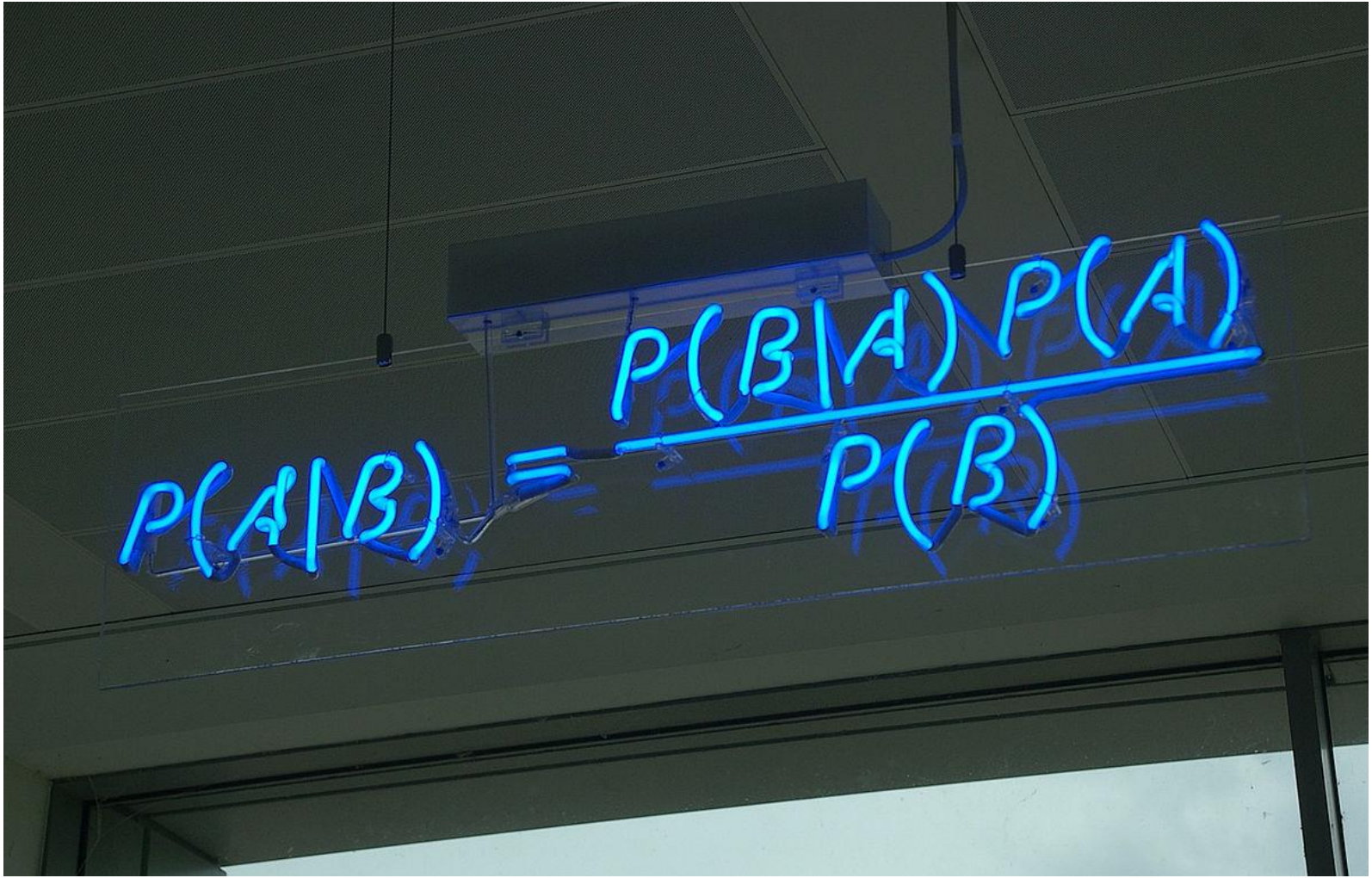
Kevin Scannell
September 10, 2018

Language modeling

- Chomsky: “… the notion of ‘probability of a sentence’ is an entirely useless one, under any known interpretation of this term.”
- Let S = “this is an entirely useless notion”
- P(S) = P(this|^) P(is|this) P(entirely|this is) … P(notion|this is an entirely useless)
- Usually formulated and computed this way (word prob. given history)
- Humans are pretty good at estimating these, at least
- P(Friday|My party is this coming) > P(Tuesday|My party is this coming)
- P(is|The man with the glasses) > P(are|The man with the glasses)

Applications

- Machine translation
- Speech recognition
- Predictive text
- Dialogue systems
- Spelling/grammar checking
- Text normalization (e.g. modernization)
- Optical character recognition
- ...


P(A|B) = P(B|A) P(A) / P(B)

Evaluation

- Extrinsic: end-to-end evaluation of more complex system
- Intrinsic: what probability does the model assign to a test corpus?
- Usually we take average negative log prob (base 2); units of “bits per word”
- 2 raised to this power is the “perplexity” of the model (PPL)
- Maximizing prob. of test corpus == minimizing bits/word or perplexity

N-gram models

- Pick an n, usually 3, 4, or 5 in practice
- First, we'll estimate $P(w_j \mid w_1 \dots w_{j-1}) \backsimeq P(w_j \mid w_{j-n+1} \dots w_{j-1})$
- Then, given a training corpus we just count things!
- $P(w_j \mid w_{j-n+1} \dots w_{j-1}) \backsimeq \#(w_{j-n+1} \dots w_j) / \#(w_{j-n+1} \dots w_{j-1})$
- The key to all of this is how to deal with zero counts
- We "smooth" these models appropriately (maybe a later talk)

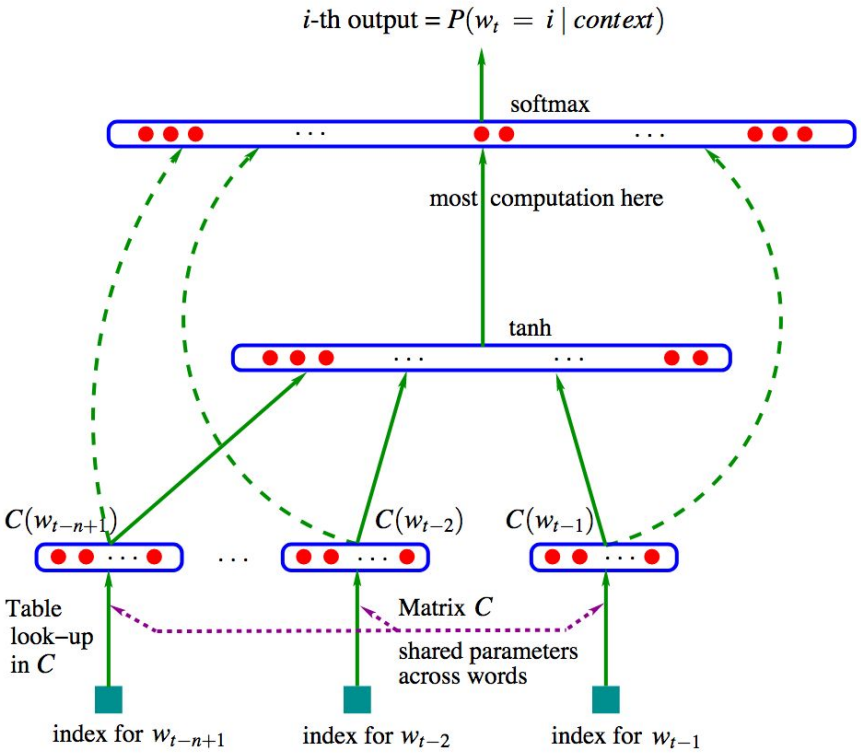
Pre-neural SotA (English!)

- Traditional test setup uses data from the Penn Treebank (PTB):
- 930k words training data, 74k dev, 82k test; 10k words in vocabulary
- 3-gram model with Kneser-Ney smoothing: PPL 148.3 (7.21 bits/word)
- 5-gram model with Kneser-Ney smoothing: PPL 141.2 (7.14 bits/word)
- Newer “billion word benchmark” (2013):
- 5-gram model with Kneser-Ney smoothing: PPL 67.6 (6.07 bits/word)

Feed-forward neural networks

- Fix a vocabulary V, each word w encoded as a d-dimensional vector C(w)
- e.g. “One-hot” encoding, dimension = size of vocabulary V (can do better!)
- NN is a device that accepts n-1 word vectors as input, outputs prob dist over V
- Original networks used one “hidden layer” of K neurons
- Each neuron accepts all d(n-1) inputs and produces a real-valued output
- Top layer has |V| neurons, each accepts input from all K hidden neurons
- Final step is a “softmax” to produce a probability distribution

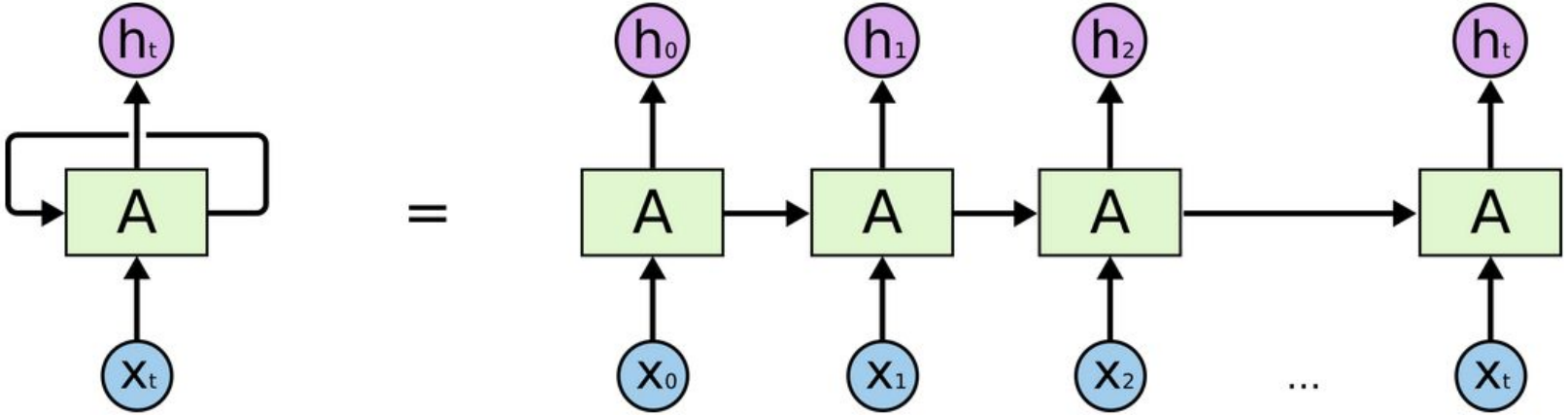
Bengio et al


i-th output = P(w_t = i | context)
softmax
most computation here
tanh
C(w_{t-n+1})
C(w_{t-2})
C(w_{t-1})
Table look-up in C
Matrix C
shared parameters across words
index for w_{t-n+1}
index for w_{t-2}
index for w_{t-1}

Training and evaluation

- Each neuron stores a number of tweakable parameters
- We train via (stochastic) gradient descent using a corpus of text
- One training example input is a sequence of n-1 words
- Output we want is the actual next word w (prob dist assigning 1.0 to w)
- Feed a training example into the network, produces a probability distribution
- Want to minimize error (or “loss”) = - log P(w)
- Compute partial derivatives of loss with respect to each parameter
- Give a small bump in the direction of the negative gradient
- Model above evaluated on PTB: PPL 141.8 (7.15 bits per word)

Recurrent neural networks



Sample Evaluations of RNN/LSTM

- “Vanilla” RNN, 400 hidden neurons on PTB: PPL 124.7 (6.96 bits per word)
- Merity et al (2017), variant of LSTM on PTB: PPL 57.3 (5.84 bits per word)
- LSTM, 8192 units evaluated on 1B benchmark: PPL 32.2 (5.01 bits per word)
- Linear interpolation of 10 LSTM models, 1B: PPL 23.7 (4.57 bits per word)

Challenges

- Implicit assumption that research on English = research on language
- Dealing with linguistic complexities (morphology)
- Lack of large training corpora for many languages
- Domain adaptation
- Dialects and non-standard varieties
- Diachronic change
- Controlling complexity / model size

Character models

- Nothing special about decomposing S into words!
- Could do syllables, morphemes, or even individual characters
- Reduces the vocabulary size significantly
- Works for languages like Chinese, Japanese, Thai, Tibetan
- Need to model longer term dependencies
- Research here has collided with work on text compression

Incorporating linguistic knowledge

- Celtic languages have “initial mutations”
- Almost always predictable from previous two words
- *bád seoil* “sailboat”, *mo b**h**ád seoil* “my sailboat”, *ár **m**bád seoil* “our sailboat”
- Word-based models don’t “see” that these are all the same word
- If most training examples are first type, say, harder to predict collocation
- Easy enough to use a *factored language model* to get better results for Irish
- Examples like this abound in Irish and other languages