Language Modeling

Overview

- What is a language model?
- How do we evaluate language models?
- Traditional language models
- Feedforward neural networks for language modeling
- Recurrent neural networks for language modeling

Overview

- What is a language model?
- How do we evaluate language models?
- Traditional language models
- Feedforward neural networks for language modeling
- Recurrent neural networks for language modeling

Language models

What is the probability of a sentence?

- Grammatically incorrect or rare sentences should be more improbable
- Or equivalently, what is the probability of a word following a sequence of words?

"The cat chased a mouse" vs "The cat chased a turnip"

Language models

What is the probability of a sentence?

- Grammatically incorrect or rare sentences should be more improbable
- Or equivalently, what is the probability of a word following a sequence of words?

"The cat chased a mouse" vs "The cat chased a turnip"

Can be framed as a sequence modeling task

Two classes of models

- Count-based: Markov assumptions with smoothing
- Neural models

Language models

What is the probability of a sentence?

- Grammatically incorrect or rare sentences should be more improbable
- Or equivalently, what is the probability of a word following a sequence of words?

"The cat chased a mouse" vs "The cat chased a turnip"

Can be framed as a sequence modeling task

Two classes of models

- Count-based: Markov assumptions with smoothing
- Neural models

We have seen this difference before. In this lecture, we will look at some details

Overview

- What is a language model?
- How do we evaluate language models?
- Traditional language models
- Feedforward neural networks for language modeling
- Recurrent neural networks for language modeling

Evaluating language models

Extrinsic evaluation

- A good language model should help with an end task such as machine translation
  - If we have a MT system that uses language models to produce outputs…
  - …a better language model can produce better outputs

Evaluating language models

Extrinsic evaluation

- A good language model should help with an end task such as machine translation
  - If we have a MT system that uses language models to produce outputs…
  - …a better language model can produce better outputs

- To evaluate a language model, is a downstream task needed?
  - Can be slow, depends on the quality of the downstream system

Evaluating language models

Extrinsic evaluation

- A good language model should help with an end task such as machine translation
  - If we have a MT system that uses language models to produce outputs…
  - …a better language model can produce better outputs

- To evaluate a language model, is a downstream task needed?
  - Can be slow, depends on the quality of the downstream system

Can we define an intrinsic evaluation?

What is a good language model?

- Should prefer good sentences to bad ones
  - It should higher probabilities to valid/grammatical/frequent sentences
  - It should assign lower probabilities to invalid/ungrammatical/rare sentences
- Can we construct an evaluation metric that directly measures this?

What is a good language model?

- Should prefer good sentences to bad ones
  - It should higher probabilities to valid/grammatical/frequent sentences
  - It should assign lower probabilities to invalid/ungrammatical/rare sentences
- Can we construct an evaluation metric that directly measures this?
  Answer: Perplexity

Perplexity

A good language model should assign high probability to sentences that occur in the real world

- Need a metric that captures this intuition, but normalizes for length of sentences

Perplexity

A good language model should assign high probability to sentences that occur in the real world

- Need a metric that captures this intuition, but normalizes for length of sentences

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$\left(P(w_1 w_2 w_3 \cdots w_n)\right)^{-\frac{1}{n}}$$

Perplexity

A good language model should assign high probability to sentences that occur in the real world

- Need a metric that captures this intuition, but normalizes for length of sentences

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$\left(P(w_1 w_2 w_3 \cdots w_n)\right)^{-\frac{1}{n}}$$

Lower perplexity corresponds to higher probability

Example: Uniformly likely words

Suppose we have n words in a sentence, and they are all independent and uniform!

- Would be a strange language….

$$\text{Perplexity} = \left(P(w_1 w_2 w_3 \cdots w_n)\right)^{-\frac{1}{n}}$$

$$= \left(\left(\frac{1}{n}\right)^n\right)^{-\frac{1}{n}} = n$$

Perplexity of history based models

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$\left(P(w_1 w_2 w_3 \cdots w_n)\right)^{-\frac{1}{n}}$$

For a history based model, we have

$$P(w_1 \cdots w_n) = \prod_i P(w_i | w_{1:i-1})$$

Perplexity of history based models

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$Perplexity = \left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}}$$

Perplexity of history based models

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$Perplexity = \left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}}$$

$$= 2^{\log_2 \left(\left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}} \right)}$$

Perplexity of history based models

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$Perplexity = \left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}}$$

$$= 2^{\log_2 \left(\left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}} \right)}$$

$$= 2^{\left(-\frac{1}{n} \sum_i \log_2 P(w_i | w_{1:i-1}) \right)}$$

Perplexity of history based models

Given a sentence $w_1 w_2 w_3 \cdots w_n$, define the perplexity of a language model as

$$Perplexity = \left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}}$$

$$= 2^{\log_2 \left(\left(\prod_i P(w_i | w_{1:i-1}) \right)^{-\frac{1}{n}} \right)}$$

$$= 2^{\left(-\frac{1}{n} \sum_i \log_2 P(w_i | w_{1:i-1}) \right)}$$

Average number of bits needed to encode the sentence

Evaluating language models

Several benchmark sets available

- Penn Treebank Wall Street Journal corpus
  - Standard preprocessing by Mikolov
  - Vocabulary size: 10K words
  - Training size: 890K tokens

- Billion Word Benchmark
  - English news text [Chelba, et al 2013]
  - Vocabulary size: ~793K
  - Training size: ~800M tokens

Standard methodology of training on the training set and evaluating on the test set

- Some papers also continue training on the evaluation set because no labels needed

Overview

- What is a language model?
- How do we evaluate language models?
- Traditional language models
- Feedforward neural networks for language modeling
- Recurrent neural networks for language modeling

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

Need to get this from data

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

$$P(w_{i+1} \mid w_{i-k:i}) = \frac{count(w_{i-k:i}, w_{i+1})}{count(w_{i-k:i})}$$

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

$$P(w_{i+1} \mid w_{i-k:i}) = \frac{count(w_{i-k:i}, w_{i+1})}{count(w_{i-k:i})}$$

The problem: Zeros in the counts.

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

$$P(w_{i+1} \mid w_{i-k:i}) = \frac{count(w_{i-k:i}, w_{i+1})}{count(w_{i-k:i})}$$

The problem: Zeros in the counts.
The solution: Smoothing

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

$$P(w_{i+1} \mid w_{i-k:i}) = \frac{count(w_{i-k:i}, w_{i+1}) + \alpha}{count(w_{i-k:i}) + \alpha|V|}$$

Many different methods for smoothing. Eg: additive smoothing, with vocabulary V

Traditional language models

Required counting n-grams

The goal: To compute $P(w_1 w_2 \cdots w_n)$ for any sequence of words

The $(k+1)^{th}$ order Markov assumption

$$P(w_1 w_2 \cdots w_n) \approx \prod_i P(w_{i+1} \mid w_{i-k:i})$$

$$P(w_{i+1} \mid w_{i-k:i}) = \frac{count(w_{i-k:i}, w_{i+1}) + \alpha}{count(w_{i-k:i}) + \alpha|V|}$$

Many different methods for smoothing. Eg: additive smoothing, with vocabulary V

The most effective non-neural smoothing method: *modified Knesser Ney smoothing*

Traditional language models

- Pros:
  - Easy to train
  - Can scale to large corpora (with careful choice of algorithms)
    - Heafield et al have written about this extensively
  - Work reasonably well

Traditional language models

- Pros:
  - Easy to train
  - Can scale to large corpora (with careful choice of algorithms)
    - Heafield et al have written about this extensively
  - Work reasonably well

- Cons:
  - Smoothing techniques are tricky to implement or modify
    - Need to implement backoff, etc
  - Scaling to large ngrams is expensive
  - Need to have seen words to generalize
    - After seeing “red ties”, “green ties”, we want to assign high probability to “blue ties”

Evaluation (perplexity)

- Penn Treebank
  - Kneser-Ney 5-gram: 140 ppl
- Billion Word Corpus
  - Kneser-Ney 5-gram: 67.6 ppl

Overview

- What is a language model?
- How do we evaluate language models?
- Traditional language models
- Feedforward neural networks for language modeling
- Recurrent neural networks for language modeling

Feedforward neural language model

[Bengio et al 2003]

- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word

Feedforward neural language model

[Bengio et al 2003]

- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word

w_1 w_2 $\cdots$ w_k

Feedforward neural language model

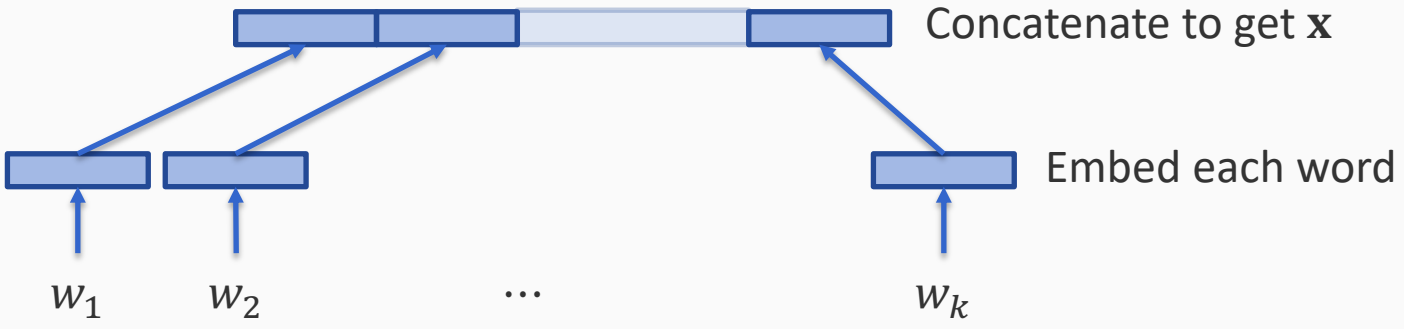
[Bengio et al 2003]

- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word

Feedforward neural language model

[Bengio et al 2003]

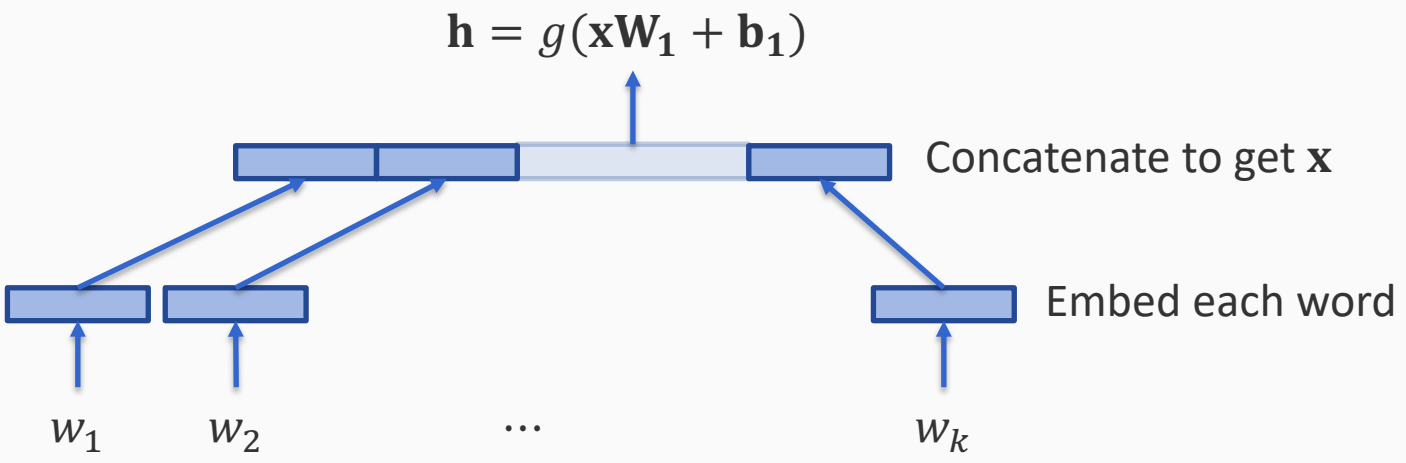
- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word



Feedforward neural language model

[Bengio et al 2003]

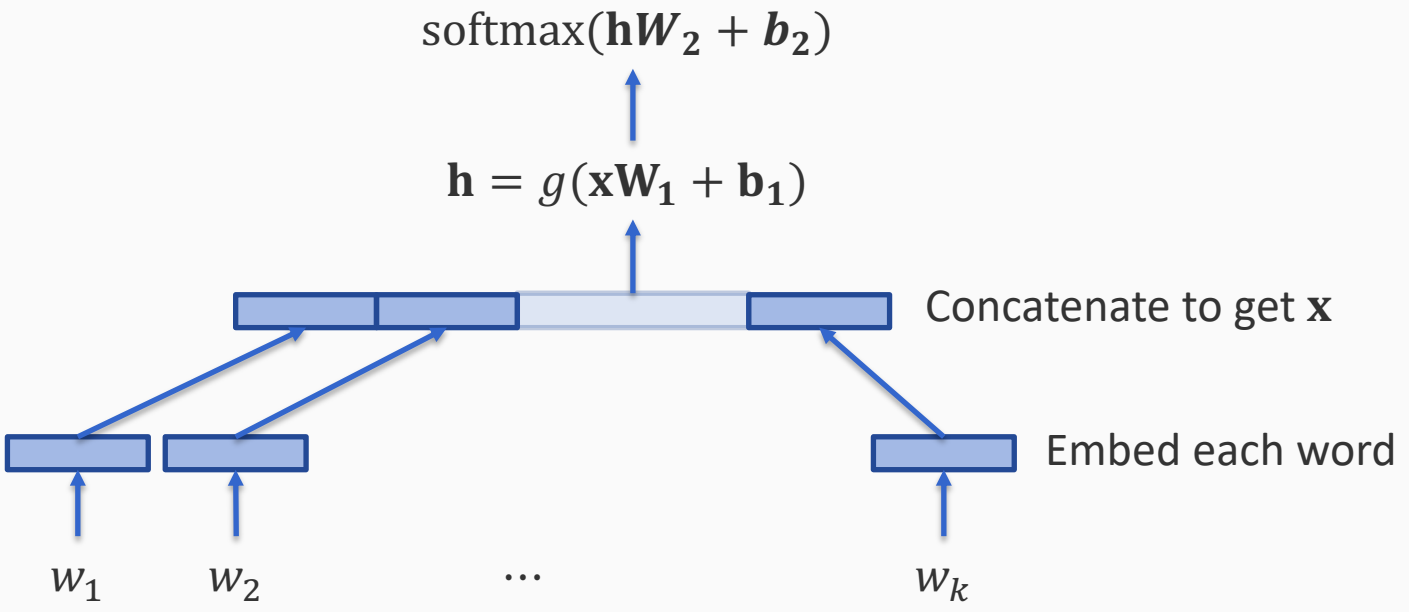
- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word



Feedforward neural language model

[Bengio et al 2003]

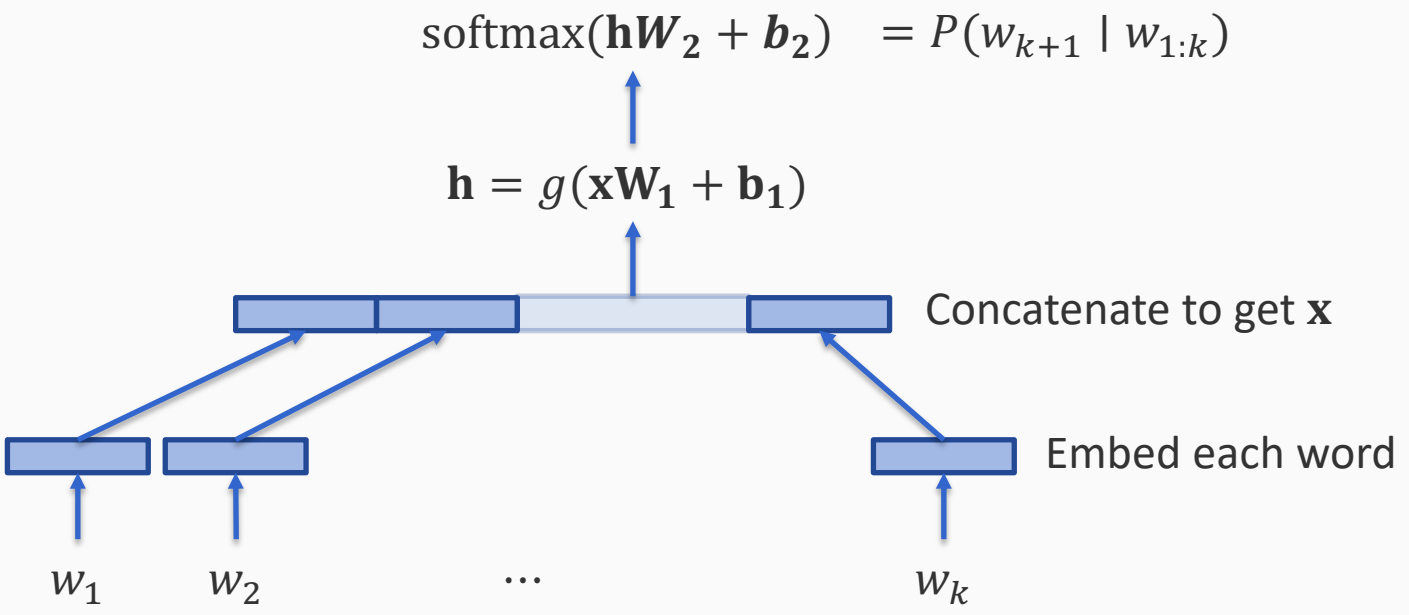
- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word



Feedforward neural language model

[Bengio et al 2003]

- Input: A sequence of k words $w_{1:k}$ in a window
- Output: A probability distribution over the next word



Feedforward neural language model

- Training data
  - K-grams from a corpus
  - Vocabulary includes all words in the training data
    - Also extra symbols for unknown words, start and end of sentences
- Trained with backpropagation
- Parameters:
  - The word embedding matrix
  - The W's and b's

Computational shortcuts

- The final softmax $\mathbf{softmax}(\mathbf{h}\boldsymbol{W_2} + \boldsymbol{b_2})$ is over the entire vocabulary
  - Can be slow

- Solutions:
  - Hierarchical softmax: An approximation that structures the softmax computation as traversing a tree with |V| nodes
    - O(log|V|) instead of O(|V|)
  - Noise contrastive estimation: Replacing the softmax with a binary classifier (as we saw with word2vec)

Feedforward neural language model

- Pros:
  - Better perplexity
  - Scales better to larger ngrams
  - Flexible architecture that admits skipgrams, etc

Feedforward neural language model

- Pros:
  - Better perplexity
  - Scales better to larger ngrams
  - Flexible architecture that admits skipgrams, etc
- Cons:
  - Computationally expensive
  - Doesn't improve translation quality over a Knesser-Ney smoothed model
    - Perhaps because it over-generalizes
    - Example: after seeing "yellow bananas" and "green bananas", it may assign a high probability to "blue bananas"
    - Rigidity of a traditional language model may be preferred

Evaluation (perplexity)

- Penn Treebank
  - Kneser-Ney 5-gram: 140 ppl
- Billion Word Corpus
  - Kneser-Ney 5-gram: 67.6 ppl
  - Hierarchical softmax + 4-gram: 101.3

Overview

- What is a language model?
- How do we evaluate language models?
- Traditional language models
- Feedforward neural networks for language modeling
- Recurrent neural networks for language modeling

Recurrent neural network language model

Starting with [Mikolov 2010-]

- We are modeling a sequence of words
  - Let us use a sequence model for this
- Can use any variant of an RNN
  - Vanilla RNN + gradient clipping [Mikolov]
  - LSTM, GRU units
- Can also include context from previous sentences or topic from the document
  - In both cases, as initial state or as part of input for each word
- We could even model a language sequence of characters
  - Or a combination

Samples from a language model

Knesser Ney 5-gram

mr. rosen contends that vaccine deficit nearby in benefit plans to take and william gray but his capital-gains provision

rural business buoyed by improved<unk>so<unk>that<unk>up<unk>progres ss pending went into nielsen visited were issued soaring searching for an equity giving

a chance affecting price after-tax legislator board closed down N cents

[Mikolov et al 2010], Penn Treebank

Samples from a language model

Knesser Ney 5-gram

mr. rosen contends that vaccine deficit nearby in benefit plans to take and william gray but his capital-gains provision

rural business buoyed by improved<unk>so<unk>that<unk>up<unk>progress pending went into nielsen visited were issued soaring searching for an equity giving

a chance affecting price after-tax legislator board closed down N cents

RNN language model

meanwhile american brands issued a new restructuring mix to<unk>from continuing operations in the west

the stock over the most results of this is very low because he could n’t develop the

peter<unk>chief executive officer says the family ariz. is left get to be working with the dollar

[Mikolov et al 2010], Penn Treebank

Samples from a language model

Knesser Ney 5-gram

mr. rosen contends that vaccine deficit nearby in benefit plans to take and william gray but his capital-gains provision

rural business buoyed by improved<unk>so<unk>that<unk>up<unk>progress pending went into nielsen visited were issued soaring searching for an equity giving

a chance affecting price after-tax legislator board closed down N cents

RNN language model

meanwhile american brands issued a new restructuring mix to<unk>from continuing operations in the west

the stock over the most results of this is very low because he could n't develop the

peter<unk>chief executive officer says the family ariz. is left get to be working with the dollar

Note: Perhaps cherry picked examples … need perplexity or extrinsic evaluations matter more

[Mikolov et al 2010], Penn Treebank

Evaluation (perplexity)

- Penn Treebank
  - Kneser-Ney 5-gram: 147.8
  - Vanilla RNN 4gram [Mikolov & Zweig 2012]: 142.1
  - Vanilla RNN 4gram + topic model [Mikolov & Zweig 2012]: 126.4
  - LSTM [Zaremba et al 2014]: 82.7
  - Variational LSTM [Gal & Ghahramani 2016]: 78.6
  - Other variants of LSTM significantly improve results:
    - AWD-LSTM + ensemble: 54.44

Evaluation (perplexity)

- Penn Treebank
  - Kneser-Ney 5-gram: 147.8
  - Vanilla RNN 4gram [Mikolov & Zweig 2012]: 142.1
  - Vanilla RNN 4gram + topic model [Mikolov & Zweig 2012]: 126.4
  - LSTM [Zaremba et al 2014]: 82.7
  - Variational LSTM [Gal & Ghahramani 2016]: 78.6
  - Other variants of LSTM significantly improve results:
    - AWD-LSTM + ensemble: 54.44
- Billion Word Corpus
  - Kneser-Ney 5-gram: 67.6
  - Hierarchical softmax + 4-gram: 101.3
  - Vanilla RNN 9gram: 51.3
  - LSTM [Jozefowicz et al 2016, Grave et al 2016]: ~43.7
  - Variants of LSTMs significantly improve perplexity
    - 10 LSTM+CNN inputs + SNM10-SKIP [Jozefowicz et al., 2016]: 23.7

Evaluation (perplexity)

- Penn Treebank
  - Kneser-Ney 5-gram: 147.8
  - Vanilla RNN 4gram [Mikolov & Zweig 2012]: 142.1
  - Vanilla RNN 4gram + topic model [Mikolov & Zweig 2012]: 126.4
  - LSTM [Zaremba et al 2014]: 82.7
  - Variational LSTM [Gal & Ghahramani 2016]: 78.6
  - Other variants of LSTM significantly improve results:
    - AWD-LSTM + ensemble: 54.44
- Billion Word Corpus
  - Kneser-Ney 5-gram: 67.6
  - Hierarchical softmax + 4-gram: 101.3
  - Vanilla RNN 9gram: 51.3
  - LSTM [Jozefowicz et al 2016, Grave et al 2016]: ~43.7
  - Variants of LSTMs significantly improve perplexity
    - 10 LSTM+CNN inputs + SNM10-SKIP [Jozefowicz et al., 2016]: 23.7

As of 2023: The best language models (in terms of perplexity) are based on transformer neural networks.

E.g. Transformer-XL Large gets 21.8 perplexity on the Billion Word Corpus

We will revisit language models after we see transformers

Examples from a Character-level RNN

Sampled one character at a time (which becomes the next input)

3 layer RNN with 512 hidden units on Shakespeare

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

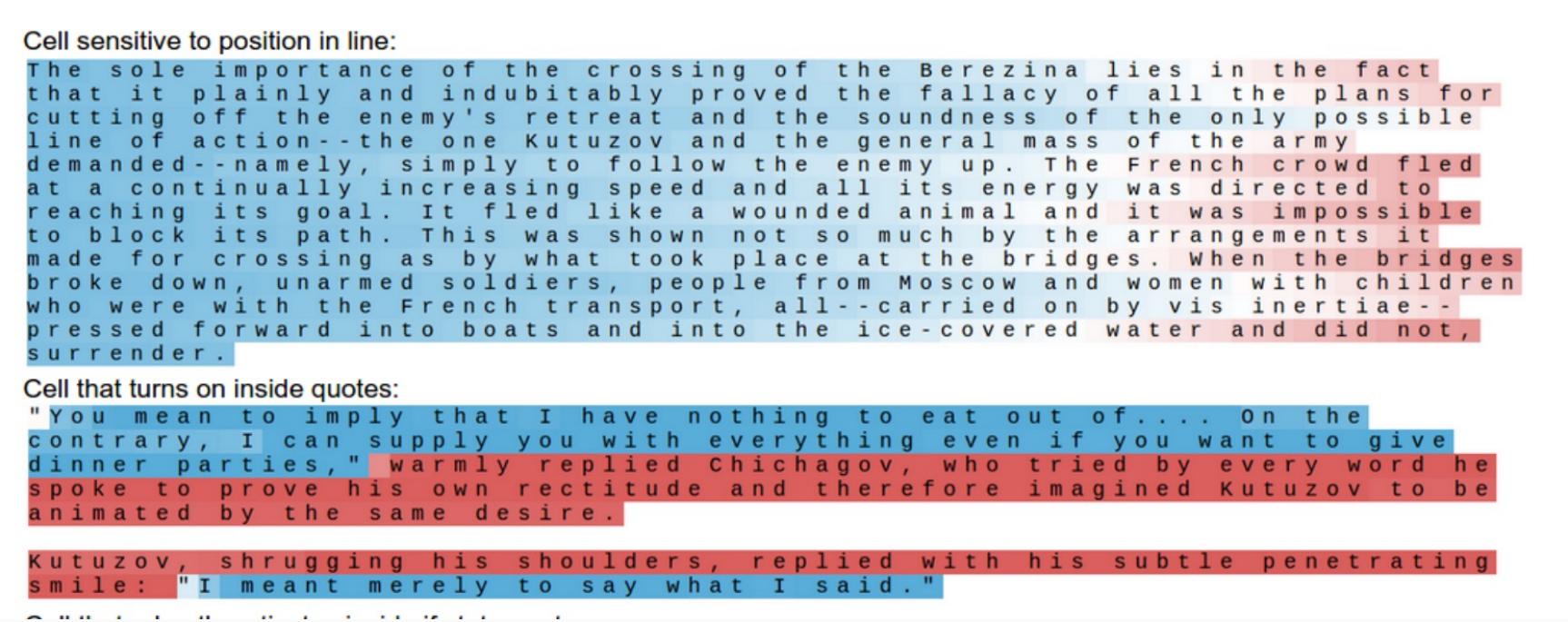
https://karpathy.github.io/2015/05/21/rnn-effectiveness/

What do we get by using an LSTM/GRU?

The hidden representation can remember where we are in the text

- Can remember different aspects of this
- Doesn’t have to remember only histories

Examples of LSTM hidden state in a language model



Karpathy, Andrej, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks." *arXiv preprint arXiv:1506.02078* (2015).

Summary: Language models

- Goal:
  - Probabilities of sentences
  - Various uses. For example, can be used to rank generated text as being valid or not
- Two broad classes of approaches
  - Traditional language model: based on counts of words in context
  - Neural language models: We saw RNNs. Today, driven by Transformers
  - Both need a lot of data to train
- Evaluated using perplexity
  - Currently, neural language models seem to be the best
- Modern language models are asked to do more than just generate words
  - They are evaluated for their ability to answer questions, chat, etc
  - We will see language models one more time after we encounter transformers