IN5550: Neural Methods in Natural Language Processing
Sub-lecture 4.3
Pre-neural language models

Andrey Kutuzov

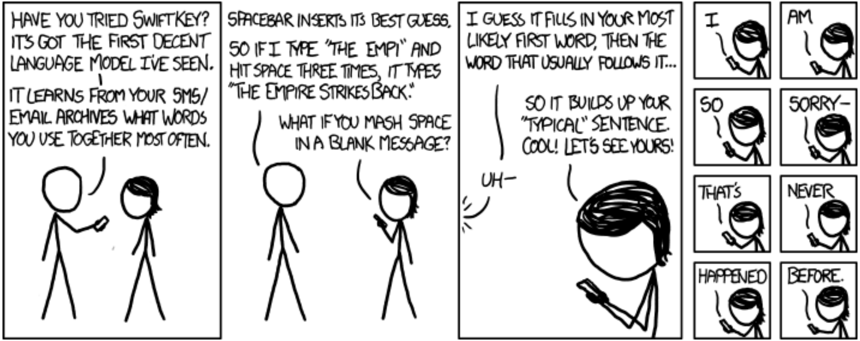
University of Oslo

14 February 2023

Contents

Language modeling (LM)

Predicting the next word in the text given the previous words:



(XKCD)

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:
  - ‘What is the probability of *lazy dog*?’

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:
  - 'What is the probability of *lazy dog*?'
  - 'What is the probability of *The quick brown fox jumps over the lazy dog*?'

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:
  - 'What is the probability of *lazy dog*?'
  - 'What is the probability of *The quick brown fox jumps over the lazy dog*?'
  - 'What is the probability of *green colorless ideas sleep furiously*?'

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:
  - 'What is the probability of *lazy dog*?'
  - 'What is the probability of *The quick brown fox jumps over the lazy dog*?'
  - 'What is the probability of *green colorless ideas sleep furiously*?'
- Task 2: to estimate the probability of a word x to follow a word sequence S of length n:

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:
  - ‘What is the probability of *lazy dog*?’
  - ‘What is the probability of *The quick brown fox jumps over the lazy dog*?’
  - ‘What is the probability of *green colorless ideas sleep furiously*?’
- Task 2: to estimate the probability of a word x to follow a word sequence S of length n:
  - ‘What is the probability of seeing *jumps* after *The quick brown fox*?’

Language modeling task definition

Modeling linguistic sequences

- Task 1: to estimate probabilities of natural language sequences:
  - 'What is the probability of *lazy dog*?'
  - 'What is the probability of *The quick brown fox jumps over the lazy dog*?'
  - 'What is the probability of *green colorless ideas sleep furiously*?'
- Task 2: to estimate the probability of a word x to follow a word sequence S of length n:
  - 'What is the probability of seeing *jumps* after *The quick brown fox*?'
- These two tasks are mathematically equivalent.

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})P(w_4|w_{1:3})...P(w_n|w_{1:n-1}) \tag{1}$$

- Any system able to yield $P(x)$ given S is a language model (LM).

Language modeling (LM)

Any language model is a text generator by definition

Language modeling (LM)

Any language model is a text generator by definition

- feed a word into the LM
- get a probability distribution over what words are likely to come next
- sample from this distribution
- feed it right back in to get the next word
- repeat this process and you're generating text!

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

Language modeling (LM)

Any language model is a text generator by definition

- feed a word into the LM
- get a probability distribution over what words are likely to come next
- sample from this distribution
- feed it right back in to get the next word
- repeat this process and you're generating text!

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

We'll see examples in this course.

Language modeling task definition

Markov assumption

- Multiplying hundreds or thousands of probabilities is not feasible practically.

Language modeling task definition

Markov assumption

- Multiplying hundreds or thousands of probabilities is not feasible practically.
- Hence, the Markov assumption a.k.a. Markov property:
  - **the future of the system is dependent only on its present state, not all the past states.**

Language modeling task definition

Markov assumption

- Multiplying hundreds or thousands of probabilities is not feasible practically.
- Hence, the Markov assumption a.k.a. Markov property:
  - **the future of the system is dependent only on its present state, not all the past states.**
- For LMs: we can take into account only the last n words to the left (n-gram models).

Markov assumption

- Multiplying hundreds or thousands of probabilities is not feasible practically.
- Hence, the Markov assumption a.k.a. Markov property:
  - **the future of the system is dependent only on its present state, not all the past states.**
- For LMs: we can take into account only the last n words to the left (n-gram models).
- It is a simplification, but it produces good results anyway.

Language modeling task definition

Markov assumption

- Multiplying hundreds or thousands of probabilities is not feasible practically.
- Hence, the Markov assumption a.k.a. Markov property:
  - **the future of the system is dependent only on its present state, not all the past states.**
- For LMs: we can take into account only the last n words to the left (n-gram models).
- It is a simplification, but it produces good results anyway.

Language modeling is widely used: text messaging, machine translation, chat-bots, summarization...., representation learning!

Evaluation of language models

'She is a researcher in natural language...

Evaluation of language models

'She is a researcher in natural language... snow-boarding'?!
I am perplexed!

Evaluation of language models

'She is a researcher in natural language... snow-boarding'?!
I am perplexed!

- One can compare LMs by their perplexity:
  - how perplexed/surprised is the model by test word sequences
  - the lower the better.

Evaluation of language models

'She is a researcher in natural language... snow-boarding'?!
I am perplexed!

- One can compare LMs by their perplexity:
  - how perplexed/surprised is the model by test word sequences
  - the lower the better.
- For each of i word tokens in the test corpus:

$$ENTROPY_i = -\log_2 LM\left(w_i \middle| w_{1:i-1}\right)$$

Evaluation of language models

‘She is a researcher in natural language... snow-boarding’?!
I am perplexed!

- One can compare LMs by their perplexity:
  - how perplexed/surprised is the model by test word sequences
  - the lower the better.
- For each of i word tokens in the test corpus:

$$\begin{aligned} ENTROPY_i &= -\log_2 LM\left(w_i \middle| w_{1:i-1}\right) \\ PERPLEXITY_i &= 2^{ENTROPY_i} \end{aligned} \tag{2}$$

Evaluation of language models

'She is a researcher in natural language... snow-boarding'?!
I am perplexed!

- One can compare LMs by their perplexity:
  - how perplexed/surprised is the model by test word sequences
  - the lower the better.
- For each of i word tokens in the test corpus:

$$\begin{aligned} ENTROPY_i &= -\log_2 LM\left(w_i | w_{1:i-1}\right) \\ PERPLEXITY_i &= 2^{ENTROPY_i} \end{aligned} \quad (2)$$

- exponentiated negative log-likelihoods per token
- For corpus perplexity, you simply average token perplexities.

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)
3. use maximum likelihood estimate for each word m from V occurring at the position i:
   $\hat{P}\left((w_i = m) \mid w_{i-2:i}\right) = \frac{\#(w_{i-2:i+1})}{\#(w_{i-2:i})}$

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)
3. use maximum likelihood estimate for each word m from V occurring at the position i:
   $\hat{P}\left((w_i = m) \,|\, w_{i-2:i}\right) = \frac{\#(w_{i-2:i+1})}{\#(w_{i-2:i})}$
4. where $\#$ are corpus counts.

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)
3. use maximum likelihood estimate for each word m from V occurring at the position i:
   $\hat{P}\left((w_i = m)\,|w_{i-2:i}\right) = \frac{\#(w_{i-2:i+1})}{\#(w_{i-2:i})}$
4. where $\#$ are corpus counts.
5. Probabilities for all seen words given previous bigrams:
   $\hat{P}\left((w_3 = jumps)\,|[brown, fox]\right) = \frac{1}{3}$

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)
3. use maximum likelihood estimate for each word m from V occurring at the position i:
   $\hat{P}\left((w_i = m)\,|w_{i-2:i}\right) = \frac{\#(w_{i-2:i+1})}{\#(w_{i-2:i})}$
4. where $\#$ are corpus counts.
5. Probabilities for all seen words given previous bigrams:
   $\hat{P}\left((w_3 = jumps)\,|[brown, fox]\right) = \frac{1}{3}$
6. because in your training corpus:

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)
3. use maximum likelihood estimate for each word m from V occurring at the position i:
   $\hat{P}((w_i = m)|w_{i-2:i}) = \frac{\#(w_{i-2:i+1})}{\#(w_{i-2:i})}$
4. where $\#$ are corpus counts.
5. Probabilities for all seen words given previous bigrams:
   $\hat{P}((w_3 = jumps)|[brown, fox]) = \frac{1}{3}$
6. because in your training corpus:
   - '*brown fox jumps*',
   - '*brown fox barks*',
   - '*brown fox barks*',

Traditional approach to LM

Old way: extract probabilities from corpus counts

1. Take a large enough corpus;
2. count all word sequences of length, say, 3;
   - (longer contexts usually better)
3. use maximum likelihood estimate for each word m from V occurring at the position i:
   $\hat{P}\left((w_i = m)\,|w_{i-2:i}\right) = \frac{\#(w_{i-2:i+1})}{\#(w_{i-2:i})}$
4. where $\#$ are corpus counts.
5. Probabilities for all seen words given previous bigrams:
   $\hat{P}\left((w_3 = jumps)\,|[brown, fox]\right) = \frac{1}{3}$
6. because in your training corpus:
   - '*brown fox jumps*',
   - '*brown fox barks*',
   - '*brown fox barks*',
7. if the previous bigram is unknown, fall back to the frequency-based method.

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;
- for the vocabulary of 10 000 words and 5-grams: 10000^5.

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;
- for the vocabulary of 10 000 words and 5-grams: 10000^5.
- Number of parameters increases exponentially with increasing k.

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;
- for the vocabulary of 10 000 words and 5-grams: 10000^5.
- Number of parameters increases exponentially with increasing k.
- Words are discrete features:

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;
- for the vocabulary of 10 000 words and 5-grams: 10000^5.
- Number of parameters increases exponentially with increasing k.
- Words are discrete features:
  - Representation power not shared between similar words

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;
- for the vocabulary of 10 000 words and 5-grams: 10000^5.
- Number of parameters increases exponentially with increasing k.
- Words are discrete features:
  - Representation power not shared between similar words
  - If we saw '*fox eats*' and '*dog eats*' 1000 times each, but never saw '*wolf eats*', the probability of '*wolf eats*' will still be 0.

Traditional approach to LM

Many shortcomings

- Sequence not seen in the training data? $\hat{P} = 0$
- There are ways to deal with unseen events...
  - but they are tricky...
  - ...and do not scale well to larger n-grams.
- Unseen events become more frequent as one increases k;
- number of possible word combinations is $|V|^k$;
- for the vocabulary of 10 000 words and 5-grams: 10000^5.
- Number of parameters increases exponentially with increasing k.
- Words are discrete features:
  - Representation power not shared between similar words
  - If we saw '*fox eats*' and '*dog eats*' 1000 times each, but never saw '*wolf eats*', the probability of '*wolf eats*' will still be 0.

Deep learning comes to help in the next sub-lecture 4.4!