Language models!

start

Feedback received from last lecture

- The '*Not examinable*' tag on some slides was not readable.
- It was not clear that the weights for CNNs were learnt through gradient descent
- You still talk too fast, please slow down, or write more slide notes
- For the debiasing, you talked about downsampling, how would you implement this?
- Please use the whiteboard for equations
- Can we please have the exercise sheet solutions sooner than mid Feb

Apologies if there was any other feedback on EdStem I may have missed

What is language modeling?

Language modeling involves assigning probabilities to sequences of words. This could involve:

1. Predicting **the next word** in a sequence of words:
   a. The next word in the sentence ....
2. Predicting a **masked word** in a sentence:
   a. The next .... in the sentence is

Why use language modelling?

Some tasks **require generating language responses**, rather than choosing a specific class....

For example:

- Word completion (on phones)
- Machine translation
- Summarization
- Copilot coding assistants
- Chatbots
- And more....

While this slide made sense 1-2 years ago, it has aged rather poorly!

Enter... ChatGPT

J

Make a gangster rap about Natural Language Processing

Yo, I'm a language model, spitting fire like it's my job
I'm here to bring the heat and keep your rhymes on the right track
I'm all about that NLP, ain't no other way
I'm here to dominate, while I generate some sweet verbiage

My experience as a researcher...

Not examinable

- **MSc**: my first paper uses LSTMs

Enter transformers

- **PhD before**: using BERT-style models for classification tasks

Enter GPT3.5 / GPT4

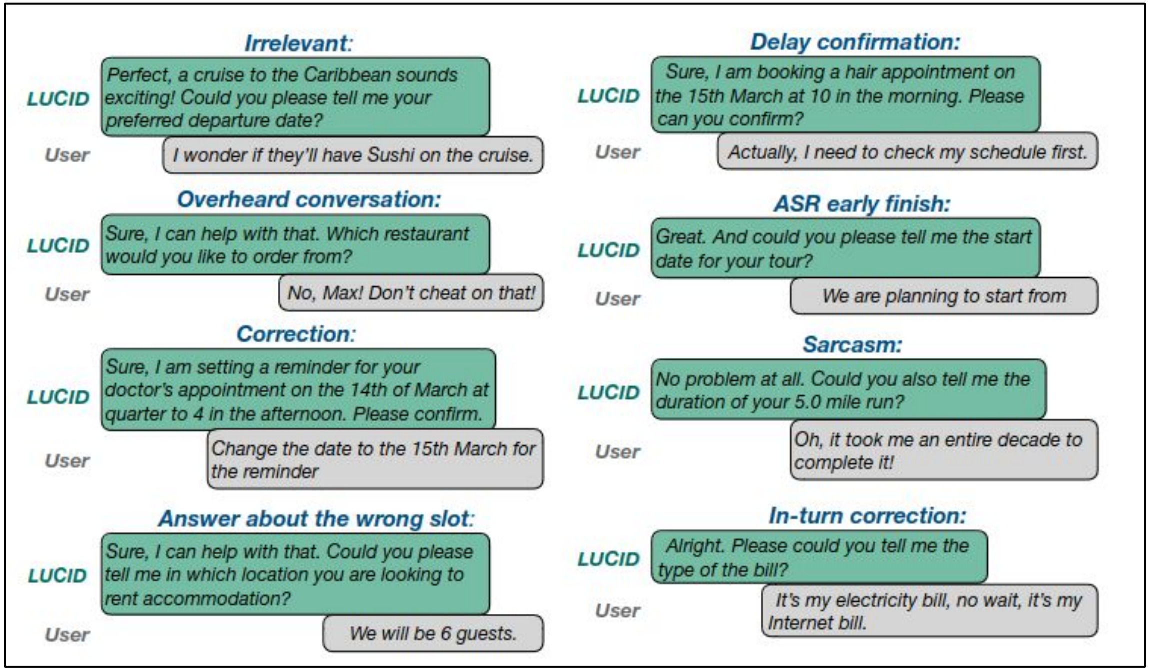
- **PhD (the last year):** Using language models to help classification models, e.g. processing data / data augmentation

- **PhD (maybe soon):** Everything with language models

My recent internship paper

Not examinable

- We wanted to create a dialogue dataset for an AI assistant.



Paper currently in review, and not yet publicly available

My recent internship paper

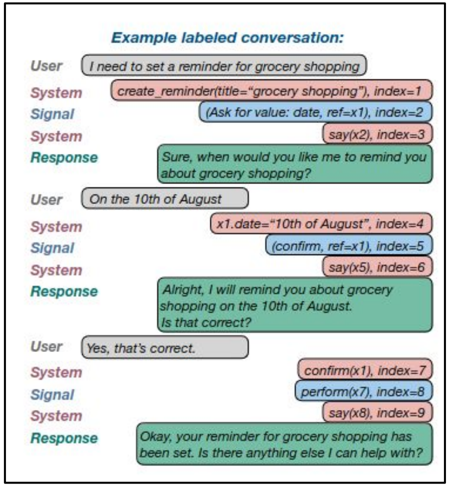
Not examinable

- We wanted to create a dialogue dataset for an AI assistant

- User commands are also annotated:

User: can you send an email to Mum, with a subject ‘birthday’
AI assistant: Sure thing! What do you want the email to say?

Command: send_email(recipient=Mum, subject=’birthday’)



Paper currently in review, and not yet publicly available

My recent internship paper

Not examinable

- We wanted to create a dialogue dataset for an AI assistant

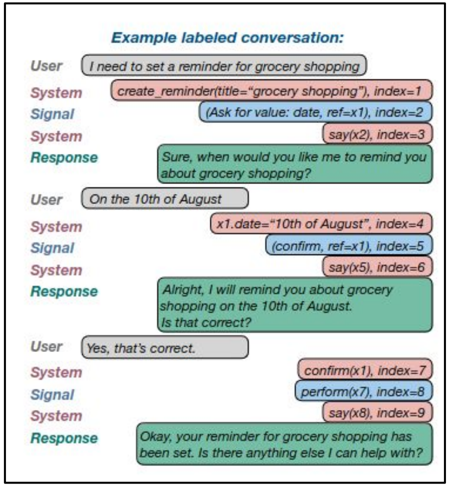
- User commands are also annotated:

User: can you send an email to Mum, with a subject ‘birthday’
AI assistant: Sure thing! What do you want the email to say?

Command: send_email(recipient=Mum, subject=’birthday’)

Let me share some challenges, and some of our solutions…

- *Good demos vs consistent reliability*
- *GPT3.5 vs GPT4*
- *Breaking down LLM calls*
- *If in doubt, discard validation*



Paper currently in review, and not yet publicly available

N-gram modelling

7 mins

Language models

We will start at the basics, with n-gram language modelling

Language models

We aim to compute $P(w|h)$ where:

- **w** is the word (or symbol) and **h** is the **history**

$$= P(w_n|w_1^{n-1})$$

Counting the likelihood of the next word

We aim to compute $P(w|h)$ where:

- **w** is the word (or symbol) and **h** is the **history**

$$= P(w_n|w_1^{n-1})$$

$$P(\text{“sat”}|\text{“the cat”}) = \frac{C(\text{“the cat sat”})}{C(\text{“the cat”})} = \frac{159,000}{116,000,000} = 0.0013$$

$$P(\text{“sat”}|\text{“the mat”}) = \frac{C(\text{“the mat sat”})}{C(\text{“the mat”})} = \frac{12,800}{17,900,000} = 0.0007$$

Counting the likelihood of the next word

- What about longer histories? Let's count google results…

$$P(\text{“weather”} \mid \text{“she wanted to find out about the”}) =$$

$$\frac{C(\text{“she wanted to find out about the weather”})}{C(\text{“she wanted to find out about the”})} = \frac{0}{74} = 0$$

N-gram models

- Assumption: N-gram models approximate history by just the few last words

$$P(\text{“weather”}\,|\,\text{“she wanted to find out about the”})$$

Full context $= P(w_n|w_1^{n-1})$

Bigram approximation $\approx P(w_n|w_{n-1}) \approx P(\text{“weather”}\,|\,\text{“the”})$

Trigram approximation $\approx P(w_n|w_{n-2}, w_{n-1})$

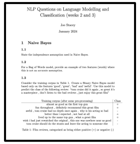
$$\approx P(\text{“weather”}\,|\,\text{“about the”})$$

N-gram models

- **Assumption**: N-gram models approximate history by just the few last words

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

Q.4.1 answered



NLP Questions on Language Modelling and Classification (weeks 2 and 3)

Joe Stacey

January 2024

1 Naive Bayes

1.1

State the independence assumption used in Naive Bayes.

1.2

For a Bag of Words model, provide an example of two features (words) where this is not an accurate assumption.

1.3

Consider the training corpus in Table 1. Create a Binary Naive Bayes model based only on the features 'good', 'great', 'bad' and 'awful'. Use this model to predict the class of the following review: "tom cruise did it again , so great it's a masterpiece , don't listen to the bad reviews , just enjoy this great film"

Training corpus (after some pre-processing)	Class
almost as good as the first top gun	+
fun throughout , definitely recommend this great film	+
awful , tom cruise had no depth once again , why is his acting so bad	-
better than i expected , not bad at all	+
lived up to the name top gun , what a great film	+
wish i had just rewatched the original , this one was nowhere near as good	-
tom cruise should do the stunts and leave the acting to someone else	-

Table 1: Film reviews, categorised as being either positive (+) or negative (-)

Extending bigrams to n-grams

- Estimating probabilities: “**MLE** as relative frequencies”

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \xrightarrow[\text{n-gram}]{\text{bigram to}} P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

- **Corpus size:**
  - The larger, the more counts - larger ***n*** possible
  - Trigrams are often enough

A worked example with n-grams

- Let’s try a worked example…..

Counting occurrences

Bi-gram language model: counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Jurafsky and Martin, Speech and Language Processing book (Chapter 3)

Counting occurrences

Let's say we want P(to | want), what do we do?

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Jurafsky and Martin, Speech and Language Processing book (Chapter 3)

Finding probabilities

Bi-gram language model: probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Question: What would each row sum to across a whole vocab?

Jurafsky and Martin, Speech and Language Processing book (Chapter 3)

Generating Shakespeare

Sampling from n-gram models trained on Shakespeare's work:

- **Unigrams**
  - Hill he late speaks; or! A more or legless first you enter

Generating Shakespeare

Sampling from n-gram models trained on Shakespeare's work:

- **Unigrams**
  - Hill he late speaks; or! A more or legless first you enter
- **Bigrams**
  - What means, sir. I confess she? Then all sorts, he is trim, captain.
  - Why doesn't stand forth they canopy, forsooth he is this palpable hit the King Henry. Live king. Follow.

Generating Shakespeare

Sampling from n-gram models trained on Shakespeare's work:

- **Unigrams**
  - Hill he late speaks; or! A more or legless first you enter
- **Bigrams**
  - What means, sir. I confess she? Then all sorts, he is trim, captain.
  - Why doesn't stand forth they canopy, forsooth he is this palpable hit the King Henry. Live king. Follow.
- **Quadrigrams**
  - King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the great banquet served in;
  - It cannot be so but so

Jurafsky and Martin, Speech and Language Processing book (Chapter 3)

Questions?

23 mins

Evaluating Language models

Introducing Perplexity

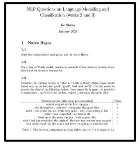
23 mins

Evaluating language models

- We can evaluate a sequence by considering the product of conditional probabilities

$$P(w_1, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

Q.4.2 answered


NLP Questions on Language Modelling and Classification (weeks 2 and 3)

Joe Stacey

January 2024

1 Naive Bayes

1.1

State the independence assumption used in Naive Bayes.

1.2

For a Bag of Words model, provide an example of two features (words) where this is not an accurate assumption.

1.3

Consider the training corpus in Table 1. Create a Binary Naive Bayes model based only on the features 'good', 'great', 'bad' and 'awful'. Use this model to predict the class of the following review: "tom cruise did it again , so great it's a masterpiece , don't listen to the bad reviews , just enjoy this great film"

Training corpus (after some pre-processing)	Class
almost as good as the first top gun	+
fun throughout , definitely recommend this great film	+
awful , tom cruise had no depth once again , why is his acting so bad	-
better than i expected , not bad at all	+
lived up to the name top gun , what a great film	+
wish i had just rewatched the original , this one was nowhere near as good	-
tom cruise should do the stunts and leave the acting to someone else	-

Table 1: Film reviews, categorised as being either positive (+) or negative (-)

Evaluating language models

Language model probabilities:

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

We now try with our bi-gram assumption:

- "I want chinese food"

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Evaluating language models

Language model probabilities:

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

We now try with our bi-gram assumption:

- “I want chinese food”
- Add <s> and </s> markers for the beginning and end of a sentence

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Evaluating language models

Language model probabilities:

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

We now try with our bi-gram assumption:

- "I want chinese food"
- Add <s> and </s> markers for beginning and end of a sentence:
- = P(<s> **I want chinese food** </s>)

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Evaluating language models

Language model probabilities:

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

We now try with our bi-gram assumption:

- “I want chinese food”
- Add <s> and </s> markers for beginning and end of a sentence:
- = P(<s> **I want chinese food** </s>)
- Assume P(i | <s>) = 0.25; P(</s> | food) = 0.68

Let’s have a try!

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

“i want chinese food”

Bi-gram language model: probabilities

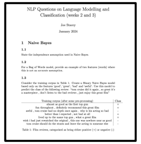
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- Remember, P(i | <s>) = 0.25; P(</s> | food) = 0.68

Jurafsky and Martin, Speech and Language Processing book (Chapter 3)

Evaluating language models

Q.4.3 answered



Language model probabilities:

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

We now try with our bi-gram assumption:

- “I want chinese food”
- Add <s> and </s> markers for beginning and end of a sentence:
- = P(<s> **I want chinese food** </s>)
- Assume P(i | <s>) = 0.25; P(</s> | food) = 0.68
- = P(i | <s>)*P(want | i)*P(chinese|want)*P(food|chinese)*P(</s>|food)
- = 0.25*0.33*0.0065*0.52*0.68

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Evaluating language models

- Multiplying many <1 numbers

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

- Switch to **log space** & replace multiplication by **addition**

$$\log\left(P(w_1, \ldots, w_n)\right) = \sum_{k=1}^{n} \log(P(w_k | w_1^{k-1}))$$

- We can then take the exponential of this

Evaluating language models

- We have an issue here with longer outputs:
    - The longer the output is, the lower its likelihood
- Our solution, **perplexity**:
    - It's the inverse probability of a text, normalized by the # of words

Evaluating language models

$$PPL(W) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$

$$= \sqrt[n]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

$$= \sqrt[n]{\frac{1}{\prod_{k=1}^{n} P(w_k | w_1^{k-1})}}$$

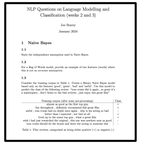
Bigram LM perplexity

$$= \sqrt[n]{\frac{1}{\prod_{k=1}^{n} P(w_k | w_{k-1})}}$$

Do we include <s> in n? How about </s>?

Evaluating language models

Q.4.5 answered



$$PPL(W) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$

$$= \sqrt[n]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

$$= \sqrt[n]{\frac{1}{\prod_{k=1}^{n} P(w_k | w_1^{k-1})}}$$

Bigram LM perplexity

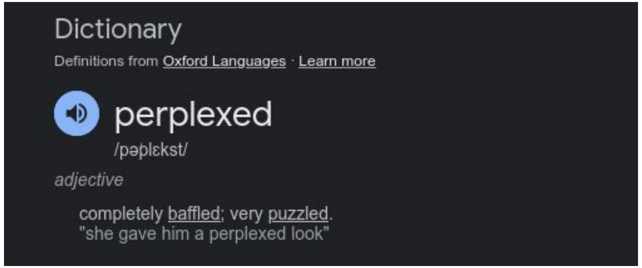
$$= \sqrt[n]{\frac{1}{\prod_{k=1}^{n} P(w_k | w_{k-1})}}$$

Do we include <s> in n? How about </s>? (no and yes)

Evaluating language models

- **Minimising** perplexity -> **maximising** probability
- It’s a measure of the surprise in a LM when seeing new text

Perplexed



Quick recap:

- For n-gram models, we create a model by counting the occurrences of n-grams in the training data

- We evaluate a model using perplexity.

- For perplexity, we take a test corpus, and predict the likelihood that our model would make made the same prediction (step by step)

About perplexity

Perplexity allows us to choose the best LM for a test data:

- **LM1 vs LM2**: the best LM is the one with the lowest perplexity

However:

- Perplexity is specific to the test-set
- How you tokenize the data matters

Questions?

35 mins

Converting between Cross Entropy Loss and Perplexity

35 mins

Cross Entropy Loss

55 mins

Cross Entropy

Cross Entropy:

- We don't know the true distribution… e.g. the likelihood of each possible next word
- We only know how many times things happen in the training data

Pytorch uses natural logarithms...

$$H(T,q) = -\sum_{i=1}^{N} \frac{1}{N} \log_e q(x_i)$$

$q(x_i)$ is the model predicted probability of the word x_i given the previous words $x_1, \ldots, x_{i-1}$

Cross Entropy

Loss for a single observation (last week, classification):

$$H(P,Q) = -\sum_i P(y_i)\log Q(y_i)$$

We choose parameters based on the loss for the whole corpus:

$$= \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} -\log p_{model}(y^{(i)}|x^{(i)};\theta)$$

Is it obvious that this follows from the definition above?

Appendix - Cross Entropy Loss (in detail)

$$H(P,Q) = -\sum_{x\in\mathcal{X}} p(x)\,\log q(x) \qquad \textbf{(Eq.1)}$$

Here x refers to (x, y)
- definition from Wikipedia

$$\arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} -\log p_{model}(y^{(i)}, x^{(i)}|\theta)$$

Sample x, y from the dataset for an approximation

$$= \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} -\log\left(p_{model}(y^{(i)}|x^{(i)};\theta)p_{model}(x^{(i)}|\theta)\right)$$

Separate to conditional probs.

$$= \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} -\log p_{model}(y^{(i)}|x^{(i)};\theta) - \log p_{model}(x^{(i)}|\theta)$$

Separate to two log expressions

$$= \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} -\log p_{model}(y^{(i)}|x^{(i)};\theta) - \log p(x^{(i)})$$

P_{model}(x) does not depend on theta

$$= \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} -\log p_{model}(y^{(i)}|x^{(i)};\theta)$$

We have our loss as we expect

Source: https://stats.stackexchange.com/questions/428937/mle-and-cross-entropy-for-conditional-probabilities

Not examinable

Cross Entropy

Cross Entropy Loss (using Pytorch):

$$H = -\sum_{i=1}^{N} \frac{1}{N} \log_{e} q(x_i)$$

$q(x_i)$ here means the predicted probability for word x_i based on the previous words (x_1, …., x_{i-1})

Converting between Cross Entropy and Perplexity

40 mins

Converting Cross Entropy to Perplexity

Converting between Perplexity and Cross Entropy:

If you calculate Cross Entropy to the base e:

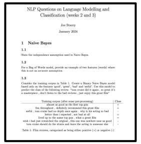
$$\text{Perplexity}(M) = e^{H}$$

If you have Cross Entropy to the base 2, this is 2^H

Converting Cross Entropy to Perplexity

Converting between Perplexity and Cross Entropy:

Q.4.6 answered



If you calculate Cross Entropy to the base e:

$$\text{Perplexity}(M) = e^{H}$$

If you have Cross Entropy to the base 2, this is 2^H

Why? Consider:

$$H = -\sum_{i=1}^{N} \frac{1}{N} \log_{e} q(x_i)$$

Questions?

50 mins

10, 20, 30 second questions about perplexity

10 second questions about perplexity

Question 1:

- If we are finding the perplexity of a single word, what is the best possible score?

10 second questions about perplexity

Question 1:

- If we are finding the perplexity of a single word, what is the best possible score?

$$\text{PPL} = \sqrt[n]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

If P(w1, .., wn) =1, we have PPL = 1

20 second questions about perplexity

Question 2:

- If our model uniformly picks words across a vocabulary of size $|V|$, what is the perplexity of a single word?

20 second questions about perplexity

Question 2:

- If our model uniformly picks words across a vocabulary of size |V|, what is the perplexity of a single word?

$$\text{PPL} = \sqrt[n]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

If n = 1, PPL = 1/p(w)

Assume p(w) = 1 / |V|
PPL = |V|

30 second questions about perplexity

Question 3:

- Our LM predicts digits between 0 and 9 as our words with even probability:
  - what is the perplexity if our test-set contains 5 words?

30 second questions about perplexity

Question 3:

- Our LM predicts digits between 0 and 9 as our words with even probability:
  - what is the perplexity if our test-set contains 5 words?

$$\text{PPL} = \sqrt[n]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

If n = 5, PPL = (1/ (p(w)*p(w)*p(w)*p(w)*p(w))^(⅕)

Assume p(w) = 1 / 10

PPL = 10

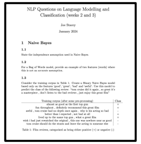
Evaluating LLMs

55 mins

Extrinsic vs Intrinsic evaluation

Q.4.7 answered

Q.4.8 answered


NLP Questions on Language Modelling and Classification (weeks 2 and 3)

1 Naive Bayes

- If the goal of your language model is to support with another task
  - The best choice of language model is the one that improves downstream task performance the most (**extrinsic** evaluation)
- Perplexity is less useful in this case (**intrinsic** evaluation)

Extrinsic vs Intrinsic evaluation

Intrinsic:

belonging to the essential nature or

intrinsic. adjective. in·trin·sic in-trin-zik, -sik. : belonging to the essential nature or constitution of a thing.

Extrinsic:

extrinsic

/ɛkˈstrɪnsɪk,ɪkˈstrɪnsɪk/

adjective

1. not part of the essential nature of someone or something; coming or operating from outside.
   "a complex interplay of extrinsic and intrinsic factors"

Using accuracy in our evaluation

- Language models, such as GPT3, can also be evaluated on their ability to perform a range of classification tasks:

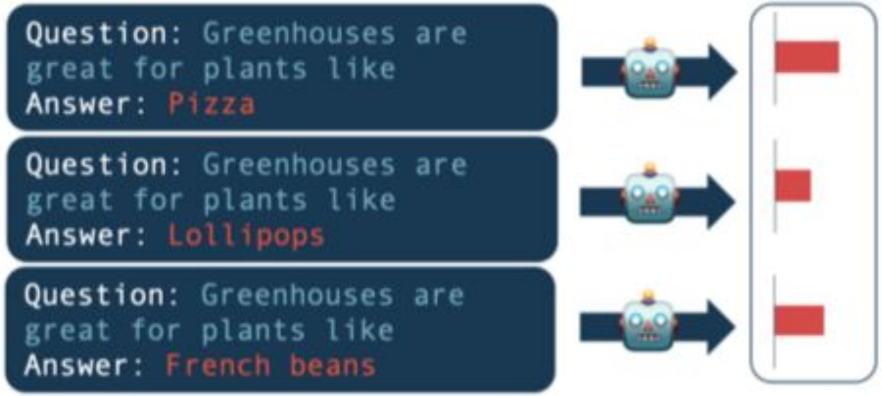


Diagram taken from:
Leveraging large language models for multiple choice question answering (Robinson et al., Nov 2022)

Using accuracy in our evaluation

- Language models, such as GPT3, can also be evaluated on their ability to perform a range of classification tasks:

Diagram taken from:
Leveraging large language models for multiple choice question answering (Robinson et al., Nov 2022)

End of lecture

60 mins

Questions?

Language models: sparsity

- WSJ corpus: built over 10 years ago
- What would happen if tested on **today's News** articles?
- What happens with unseen n-grams?
  - E.g. "His Majesty" or "Trussonomics"

Sparsity

<UNK>

Language models: sparsity

- Techniques to mitigate sparsity:
    - Add-1 Smoothing
    - Back-off
    - Interpolation

Language models: sparsity

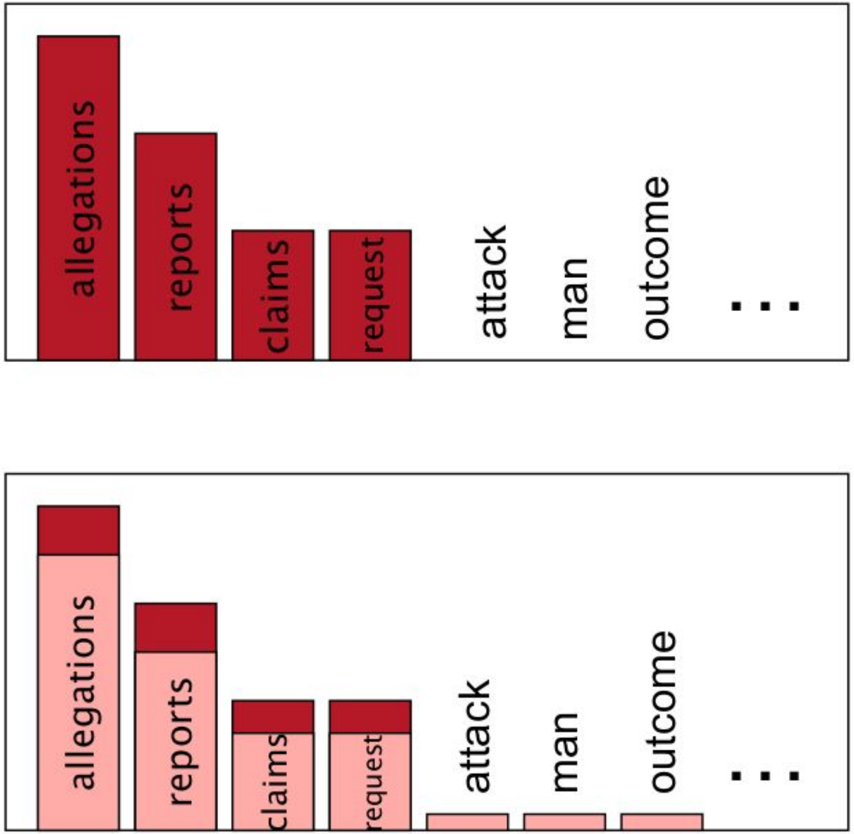
Add-one smoothing

- Given words with sparse statistics, steal probability mass from more frequently words
    - Better generalization

Bigram example

$$P_{add-1}(w_n|w_{n-1}) = \frac{C(w_{n-1},w_n)+1}{C(w_{n-1})+V}$$

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)



Language models: sparsity

- Bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Language models: sparsity

- Bigram counts

$$C(\text{“to eat”}) = 686$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Language models: sparsity

- Bigram **add-1** counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	**1**	10	**1**	**1**	**1**	3
want	3	**1**	609	2	7	7	6	2
to	3	**1**	5	687	3	**1**	7	212
eat	**1**	**1**	3	**1**	17	3	43	**1**
chinese	2	**1**	**1**	**1**	**1**	83	2	**1**
food	16	**1**	16	**1**	2	5	**1**	**1**
lunch	3	**1**	**1**	**1**	**1**	2	**1**	**1**
spend	2	**1**	2	**1**	**1**	**1**	**1**	**1**

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Language models: sparsity

- Bigram **add-1** smoothed estimates

$$P_{add-1}(w_n|w_{n-1}) = \frac{C(w_{n-1},w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	**0.00025**	0.0025	**0.00025**	**0.00025**	**0.00025**	0.00075
want	0.0013	**0.00042**	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	**0.00026**	0.0013	0.18	0.00078	**0.00026**	0.0018	0.055
eat	**0.00046**	**0.00046**	0.0014	**0.00046**	0.0078	0.0014	0.02	**0.00046**
chinese	0.0012	**0.00062**	**0.00062**	**0.00062**	**0.00062**	0.052	0.0012	**0.00062**
food	0.0063	**0.00039**	0.0063	**0.00039**	0.00079	0.002	**0.00039**	**0.00039**
lunch	0.0017	**0.00056**	**0.00056**	**0.00056**	**0.00056**	0.0011	**0.00056**	**0.00056**
spend	0.0012	**0.00058**	0.0012	**0.00058**	**0.00058**	**0.00058**	**0.00058**	**0.00058**

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Language models: sparsity

- Compared to original (not smoothed) version

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Jurafsky and Martin, Speech and Language Processing Boko (Chapter 3)

Language models: sparsity

- Compared to original (not smoothed) version

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Summary of Add-1 smoothing

- Easy to implement
- But takes too much probability mass from more likely occurrences
- Assigns too much probability to unseen events
- Could try +k smoothing with a smaller value of k

Some slides for Monday's lecture...

Back off smoothing

- If we do not have any occurrences of a ‘his royal highness’:
    - We could **back-off** and see how many occurrences there are of ‘royal highness’

Back off smoothing ("stupid back-off")

- If we do not have any occurrences of 'you had covid':

$$S(w_i|w_{i-2}\,w_{i-1}) = \begin{cases} \dfrac{C(w_{i-2}\,w_{i-1}\,w_i)}{C(w_{i-2}\,w_{i-1})} & \text{if } C(w_{i-2}\,w_{i-1}\,w_i) > 0 \\ 0.4 \cdot S(w_i|w_{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i|w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}\,w_i)}{C(w_{i-1})} & \text{if } C(w_{i-1}\,w_i) > 0 \\ 0.4 \cdot S(w_i) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{C(w_i)}{N}$$

From: *Marek Rei, Machine Learning for Language Modelling part 2.*

Wikipedia tells us:

“This model generally works well in practice, but fails in some circumstances.

For example, suppose that the bigram "a b" and the unigram "c" are very common, but the trigram "a b c" is never seen. Since "a b" and "c" are very common, it may be significant (that is, not due to chance) that "a b c" is never seen.

Perhaps it's not allowed by the rules of the grammar. Instead of assigning a more appropriate value of 0, the method will back off to the bigram and estimate $P(c \mid b)$, which may be too high”

Interpolation

- We combine evidence from different n-grams:

$$\begin{aligned} P_{interp}(w_i|w_{i-2}\ w_{i-1}) = \quad & \lambda_1 P(w_i|w_{i-2}\ w_{i-1}) \\ & +\lambda_2 P(w_i|w_{i-1}) \\ & +\lambda_3 P(w_i) \end{aligned}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

From: *Marek Rei, Machine Learning for Language Modelling part 2.*

Questions?

Language models: evaluation

- Train LM on 38 million words of WSJ
  - Test on 1.5 million held-out words also from WSJ

Unigram PPL	Bigram PPL	Trigram PPL
962	170	109

PPL is short for perplexity

Discussion

- N-gram LM: good approximation of language likelihood
- However, even with larger n (say 4-5), n-gram language models fail to model long-distance dependencies, e.g.:

> "**The GPU machines** which I had just bought from a reputable supplier and put in the server room in the other building **crashed**."

Solution?

Questions?

1hr 45 mins

Neural Language Models

- Neural-based LMs have several improvements:
  - Avoids n-gram sparsity issue
  - Contextual word representations i.e. **embeddings**

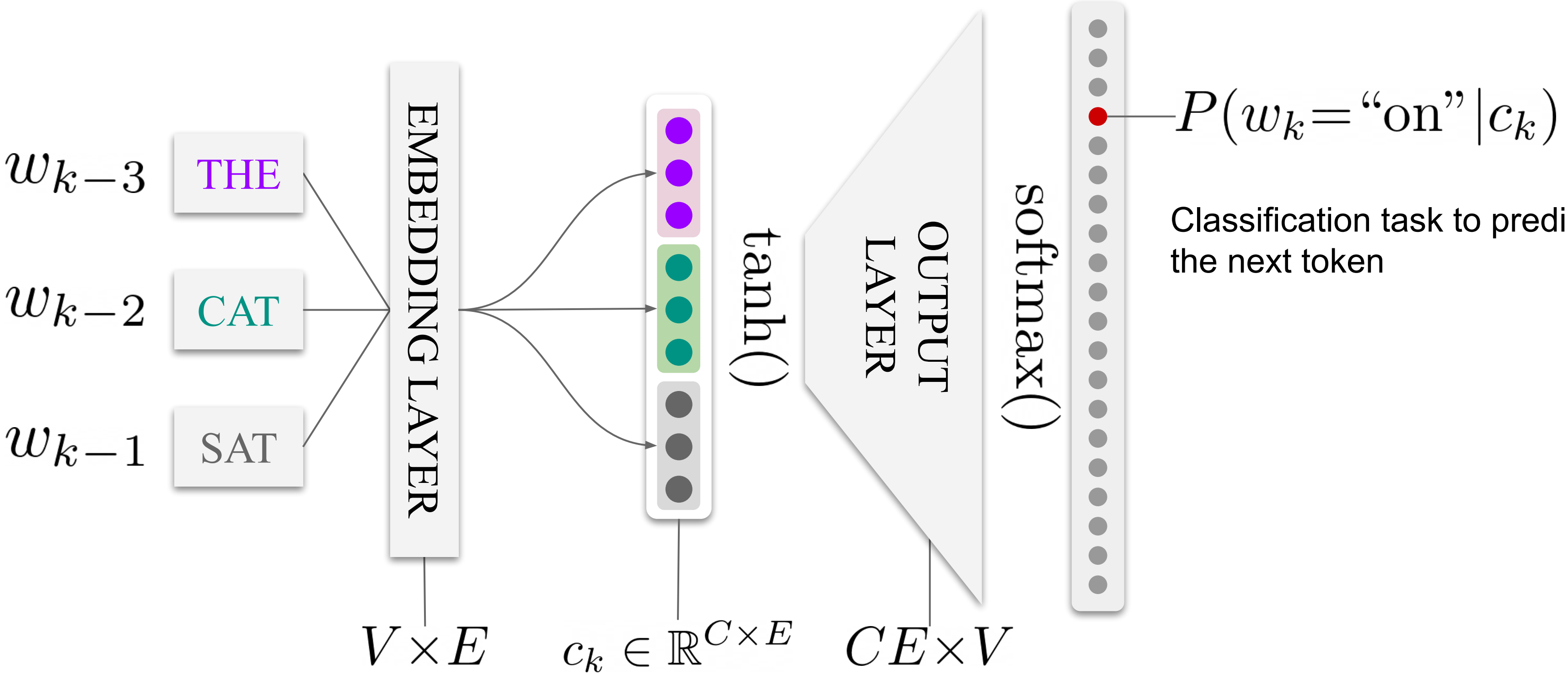
Neural Language Models (NLM)

$$P(w_1, \ldots, w_n) = \prod_{k=1}^{n} \underbrace{P(w_k |}_{\text{A classification task}} \underbrace{w_1^{k-1})}_{\text{conditioned on history}}$$

which repeats through “time”

- **Idea**: Use a DNN to model
  - Feed-forward LMs
  - Recurrent LMs

4-gram Feed-forward LM (FFLM)



Classification task to predict the next token

[A Neural Probabilistic Language Model (Bengio et al. 2003)](#)

Feed-forward LM (FFLM)

- **First applications of DNNs to LM**
  - Approximates history with last C words
    - C affects model size!

- **Ex:** 4-gram FFLM has a context size of 3
  - Estimates
  - The context is formed by concatenating word embeddings

$$c_k = [\textsc{Emb}(\text{“the”}); \textsc{Emb}(\text{“cat”}); \textsc{Emb}(\text{“sat”})]$$

Feed-forward LM (FFLM)

- **First successful attempt to neural LMs**
  - Simple and elegant NN perspective to n-gram LMs
  - 10 to 20% perplexity improvement over smoothed 3-gram LM (Bengio et al. 2003)

- **Quickly superseded by more expressive RNN LMs**

Questions?

Perplexity and Cross Entropy

Change of Base Rule

$$\log_b a = \frac{\log_x a}{\log_x b}$$

Let b = 2, a = q(x), x = e

So we divide by $\log_e 2$ to convert CELoss in base e to base 2