Foundations of Language Modeling

COM6513/4513: Natural Language Processing

Nafise Sadat Moosavi

Computer Science Department

Readings:
Chapter 3 and Chapter 7 from from
Jurafsky and Martin
Chapter 6 from Eisenstein

Attendance Monitoring

BD-WM-DF

Questions?

Recap

If two words never appear in exactly the same sentences, how can embeddings still learn that they are similar?

Recap

Embedding E_1 achieves a Spearman correlation of 0.85 on WordSim-353.
Embedding E_2 achieves a Spearman correlation of 0.73 on WordSim-353.

You plan to use one of these embeddings in a named entity recognition (NER) system. Which embedding will perform better on the NER task?

Lecture Plan

- What is Language Modeling?
- The Language Modeling Pipeline
- Statistical Language Models
- Evaluation & Practical Issues
- Tokenization

What is a Language Model?

A language model assigns probabilities to sequences: $P(w_1, w_2, ..., w_n)$

Or equivalently: $P(w_t | w_1, ..., w_{t-1})$

It answers:

“What comes next?”

Applications?

Applications

Language models power:

- Search query completion
- Machine translation
- Chatbots
- Code generation
- Foundation models

Modern AI is largely language modeling at scale.

The Language Modeling Pipeline

Language Modeling System =

1. 📚 Training Data (Corpora)
2. 🔤 Input Representation (Tokenization)
3. Model Architecture
4. ⚙️ Optimization (Training Objective)
5. 📏 Evaluation
6. 🚀 Deployment / Generation

Training Data

Pretraining =

Train a language model on a very large corpus using a general objective (e.g., next token prediction).

Then:

- Fine-tune for downstream tasks (Only if needed)

What Makes Good Pretraining Data?

What Makes Good Pretraining Data?

- Scale (billions/trillions of tokens)
- Diversity (domains, styles)
- Cleanliness
- Language coverage
- Deduplication
- Filtering toxic/low-quality content

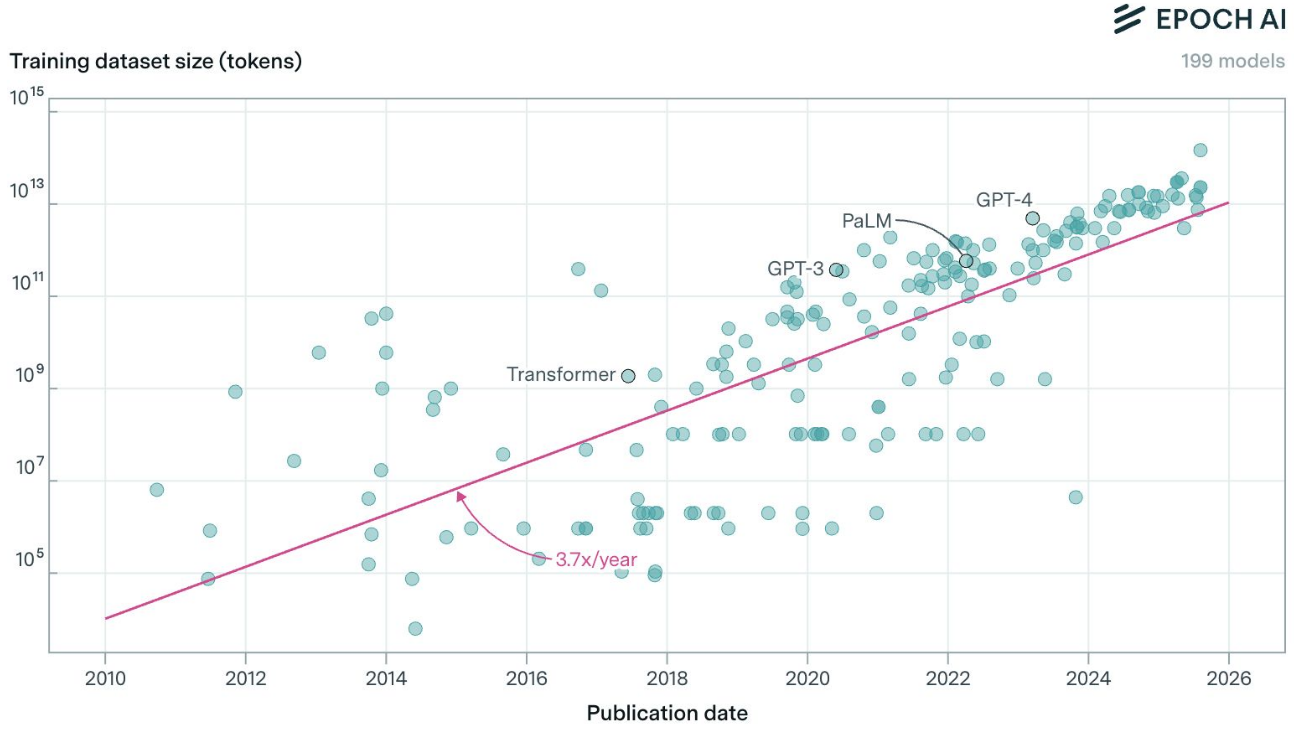
Typical Sources of Pretraining Data

- Web crawl (Common Crawl)
- Wikipedia
- Books
- News
- Code repositories
- Academic articles
- Forums (e.g., StackOverflow)

Training Data

The size of datasets used to train language models doubles approximately every six months

Source: https://epoch.ai/data-insights/dataset-size-trend



Data Scale and Performance

Modern scaling observation:

Performance improves with:

- More parameters
- More data
- More compute

Data Curation Challenges?

Data Curation Challenges

- Copyright
- Bias
- Toxicity
- Misinformation
- Multilingual imbalance

Pretraining vs Fine-tuning

Pretraining vs Fine-tuning

- Massive data
- General objective
- Expensive
- Learns language structure

- Smaller labeled dataset
- Task-specific objective
- Cheaper
- Learns task behavior

The Language Modeling Pipeline

Language Modeling System =

1. 📚 Training Data (Corpora)
2. 🔤 Input Representation (Tokenization)
3. Model Architecture
4. ⚙️ Optimization (Training Objective)
5. 📏 Evaluation
6. 🚀 Deployment / Generation

What is a Language Model?

A language model assigns probabilities to sequences: $P(w_1, w_2, \ldots, w_n)$

Or equivalently: $\boxed{P(w_t | w_1, \ldots, w_{t-1})}$

It answers:

“What comes next?”

Historical Evolution

- Count-based models
  - N-grams
  - Explicit probability estimation
  - Data sparsity & smoothing
- Neural LMs
  - Feedforward LMs
  - RNNs / LSTMs
  - Distributed representations
- Transformer-based LMs
- Scaled Transformer Models (LLMs)

Historical Evolution

- Count-based models
- Neural LMs
- Transformer-based LMs
    - Self-attention
    - Parallelization
    - Long-range dependency modeling
- Scaled Transformer Models (LLMs)
    - Massive data
    - Massive parameters
    - Emergent abilities
    - In-context learning

Language Modeling

Given a text corpora, how would you compute P(Greenest| Sheffield is the)?

Language Modeling

P(Greenest| Sheffield is the) = Count (Sheffield is the greenest)/ Count (Sheffield is the)

Chain Rule

We reduce joint probability to conditional probabilities.

$$\begin{aligned} P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2})\ldots P(X_n|X_{1:n-1}) \\ &= \prod_{k=1}^{n} P(X_k|X_{1:k-1}) \end{aligned}$$

Chain rule of probability: decomposing complex prob into smaller, more manageable conditional ones

From Probability to Log-Likelihood

Why Raw Probability Is Problematic

For long sequences:

$$P(X_{1:n}) = \prod_{k=1}^{n} P(X_k \mid X_{1:k-1})$$

- Multiplying many small numbers
- Extremely small values
- Hard to compare across sentences of different lengths

From Probability to Log-Likelihood

- Numerical Stability (Avoid Underflow)

  Sums of negative numbers are numerically stable

- Computational Efficiency

  Addition is computationally cheaper and more stable

$$\log P(X_{1:n}) = \log \left(\prod_{k=1}^{n} P(X_k \mid X_{1:k-1}) \right)$$

$$\log P(X_{1:n}) = \sum_{k=1}^{n} \log P(X_k \mid X_{1:k-1})$$

Statistical Language Models

N-gram Assumption

Instead of full history:

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

Markov assumption

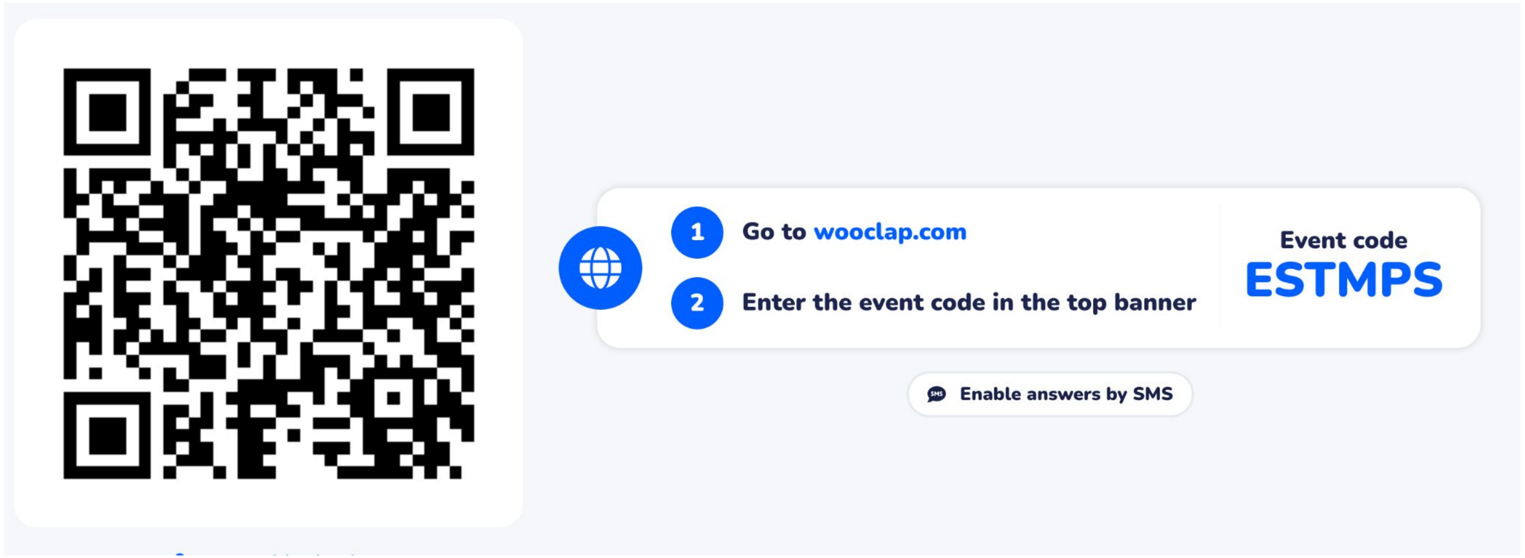
Bigram LM

Ice cream is delicious

<s> ice, Ice cream, cream is, is delicious, delicious </s>

Bigram LM

P(Greenest| Sheffield is the) =?



1 Go to wooclap.com

2 Enter the event code in the top banner

Event code
ESTMPS

Enable answers by SMS

Bigram LM

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k | w_{k-1})$$

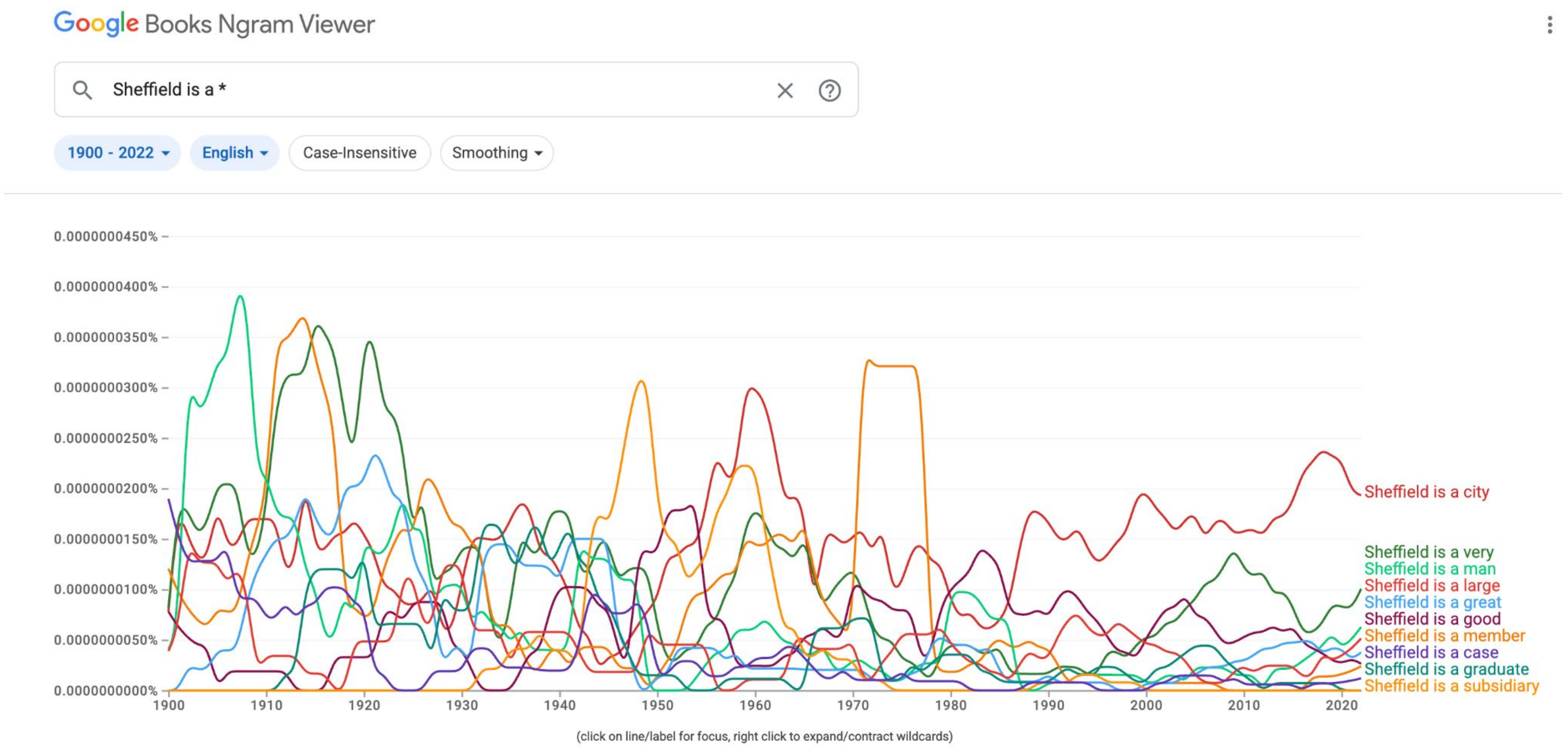
Bigram LM

$$P(w_n|w_{n-1}) = \text{Count}(w_{n-1}w_n) / \text{Count}(w_{n-1})$$

N-gram LM

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\;w_n)}{C(w_{n-N+1:n-1})}$$

Example



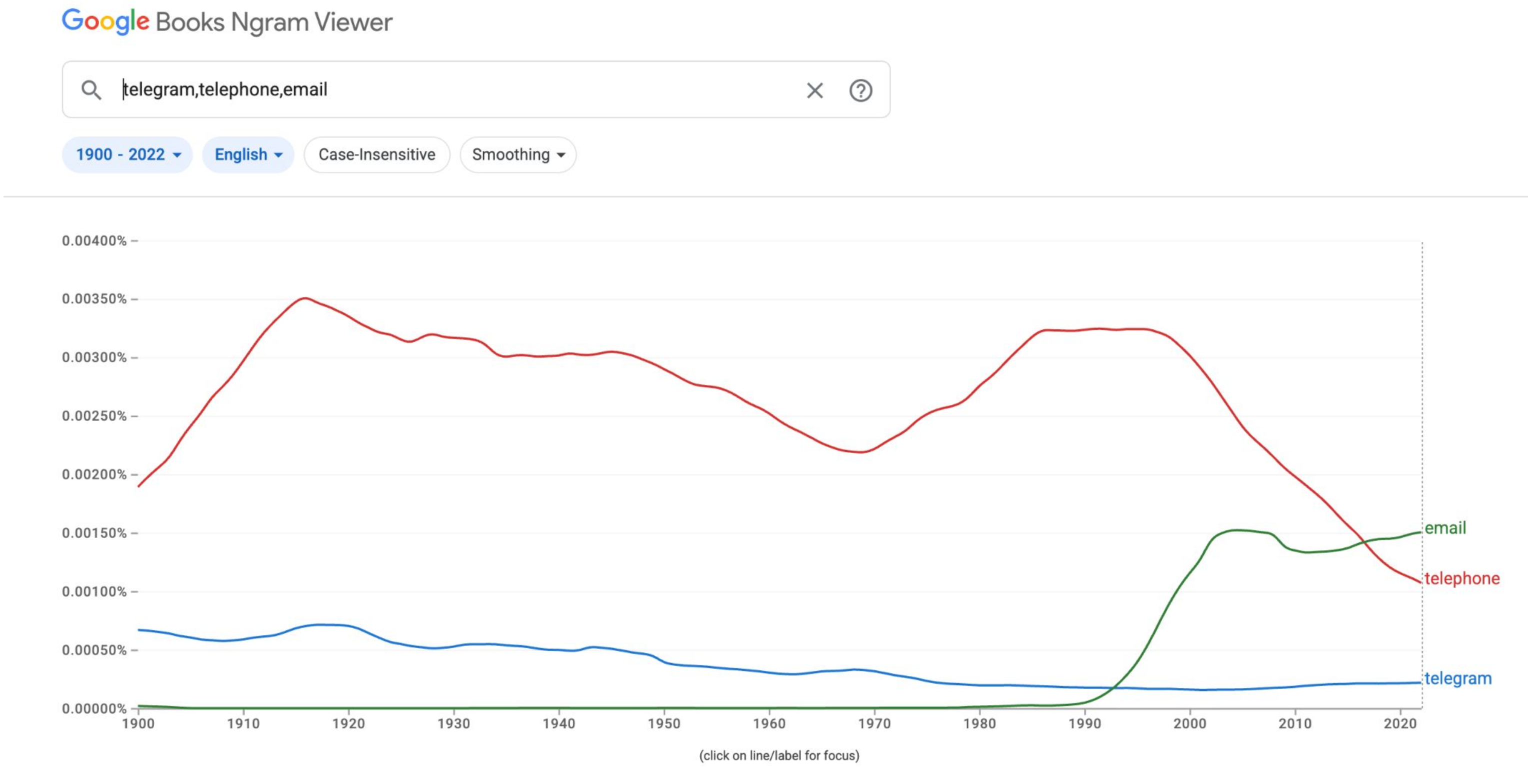
Google Books Ngram Viewer

Sheffield is a *

1900 - 2022 English Case-Insensitive Smoothing

(click on line/label for focus, right click to expand/contract wildcards)

Example

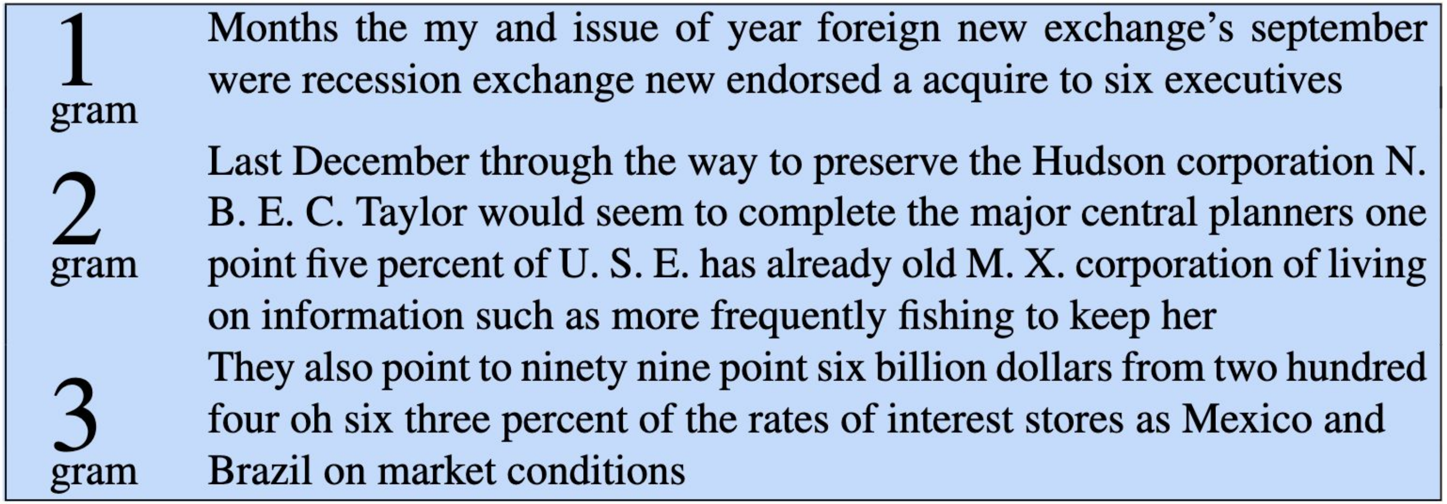


Google Books Ngram Viewer

telegram,telephone,email

1900 - 2022 ▾ English ▾ Case-Insensitive Smoothing ▾

Example



1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Figure 3.5 Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

Source: https://web.stanford.edu/~jurafsky/slp3/3.pdf

Strengths of N-grams

- Simple
- Interpretable
- Efficient
- Strong baseline

Weaknesses

- Data sparsity
- Exponential growth of parameters
- No generalization across similar words
- Zero probability problem
- Fixed length context
- Unknown Words

Weaknesses

- Data sparsity
- Exponential growth of parameters
- No generalization across similar words
- Zero probability problem
- Fixed length context
- Unknown Words

For a 3-gram LM with vocabulary size V = 2000, how many parameters does the model require?

Weaknesses

- Data sparsity
- Exponential growth of parameters
- $\boxed{\text{No generalization across similar words}}$
- Zero probability problem
- Fixed length context
- Unknown Words

Smoothing

What happens if seen words appear with an unseen context in test data?

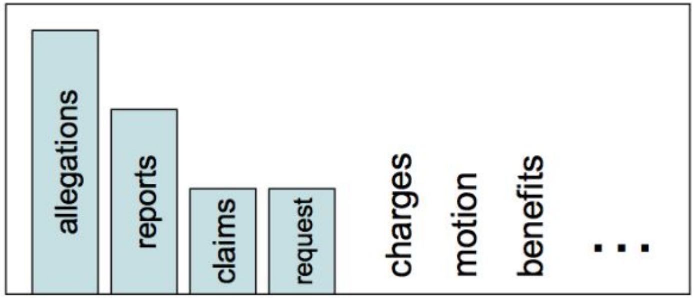
✓ Smoothing (discounting)

- Stealing from the rich and giving to the poor!
- Reducing the probability of some more frequent events
- Giving it to the unseen events

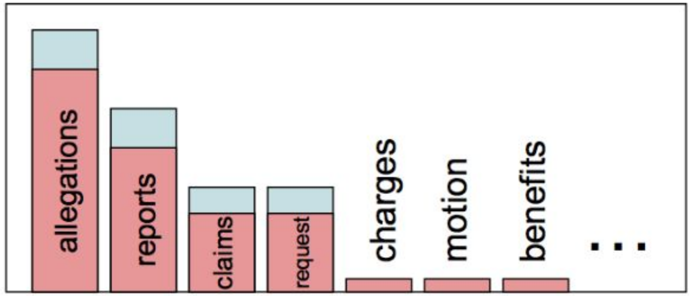
Smoothing

Smoothing flattens spiky distributions to make them generalize better

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
7 total



Source: https://people.eecs.berkeley.edu/~klein/cs288

The Language Modeling Pipeline

Language Modeling System =

1. 📚 Training Data (Corpora)
2. 🔤 Input Representation (Tokenization)
3. Model Architecture
4. ⚙️ **Optimization (Training Objective)**
5. 📏 Evaluation
6. 🚀 Deployment / Generation

The Language Modeling Pipeline

Language Modeling System =

1. 📚 Training Data (Corpora)
2. 🔤 Input Representation (Tokenization)
3. Model Architecture
4. ⚙️ **Optimization (Training Objective)**
5. 📏 Evaluation
6. 🚀 Deployment / Generation

Optimization = Maximum Likelihood Estimation (MLE)

Maximize the likelihood of the training corpus

The Language Modeling Pipeline

Language Modeling System =

1. 📚 Training Data (Corpora)
2. 🔤 Input Representation (Tokenization)
3. Model Architecture
4. ⚙️ Optimization (Training Objective)
5. 📏 **Evaluation**
6. 🚀 Deployment / Generation

Evaluation

- Intrinsic
  - Perplexity
- Extrinsic
  - Downstream performance
  - comparing performances of an ASR model using two different LMs

Intrinsic Evaluation

- Creating a test set which in independent of the training and development set
- How to evaluate which LM is better on the test set?

Intrinsic Evaluation

- Creating a test set which in independent of the training and development set
- Which model assigns a higher probability to the test set

Perplexity

Inverse probability of the test set normalized by the number of words

$$
\begin{aligned}
\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}
\end{aligned}
$$

Perplexity

Inverse probability of the test set normalized by the number of words

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

Lower better or higher?

Perplexity in Log Form

$$\log \mathrm{PP}(W) = \log \left(P(w_1 \ldots w_N)^{-\frac{1}{N}} \right) \quad = -\frac{1}{N} \log P(w_1 \ldots w_N)$$

$$\log \mathrm{PP}(W) = -\frac{1}{N} \sum_{k=1}^{N} \log P(w_k \mid w_{1:k-1})$$

$$\mathrm{PP}(W) = \exp \left(-\frac{1}{N} \sum_{k=1}^{N} \log P(w_k \mid w_{1:k-1}) \right)$$

Question

If a model predicts every word uniformly over a vocabulary of size V, what is its perplexity?

Question

Suppose two models assign exactly the same conditional probabilities for each word, but one test set is twice as long as the other.

Will perplexity change?

Relation to Cross-Entropy

- Measuring how surprised the model is by the test data
- If the model assigns high probability to correct words
  - Log probabilities are closer to 0
  - Negative average is smaller
  - Cross-entropy is lower

$$H = -\frac{1}{N}\sum_{k=1}^{N}\log P(w_k \mid w_{1:k-1})$$

Lower cross-entropy = better predictions

Relation to Cross-Entropy

- Minimize cross-entropy
- ⇔ maximize normalized log-likelihood
- ⇔ minimize perplexity

$$H = -\frac{1}{N}\sum_{k=1}^{N} \log P(w_k \mid w_{1:k-1})$$

Perplexity = exp (cross-entropy)

Special Cases

Perfect Model

- $H = 0$
- $PP = 1$
- No uncertainty

Uniform Distribution Over V Words

- $P(w_k)=1/V$
- $PP = V$
- Maximum uncertainty

Interpreting Perplexity

Can be interpreted as effective average branching factor

If PP = 100
→ Model behaves as if choosing among 100 equally likely words.

If PP = 10
→ Model is much more confident.

If PP = 1
→ Perfect prediction.

A Hidden Assumption So Far

We treated each word as a symbol

$$P(w_k \mid w_{1:k-1})$$

Implicit assumptions:

- Finite vocabulary
- Each word is atomic
- Words are the basic prediction units

Perplexity Depends on Tokenization

Perplexity is measured per token

If we change tokenization:

- Change N
- Change the prediction space
- Change perplexity

$$\mathrm{PP} = \exp\left(-\frac{1}{N}\sum \log P(\mathrm{token})\right)$$

Evaluation depends on representation

The Out-of-Vocabulary (OOV) Problem

Vocabulary = words seen in training

Unseen word → probability = 0

Perplexity=∞

Unknown Words

- Using a special <UNK> token
    - Choose a fixed vocabulary in advance.
    - Convert words in the training data that doesn’t appear in the vocabulary to <UNK>
    - Estimate the probabilities for <UNK> similar to other words

Why is Handling New Words Hard?

- Each word has a fixed vector
- Word that are not in the training data → No representation
- Seeing "happiness" will not help in representing "unhappiness"
- We need a better way to represent unseen words dynamically

The Language Modeling Pipeline

Language Modeling System =

1. 📚 Training Data (Corpora)
2. 🔤 Input Representation (Tokenization)
3. Model Architecture
4. ⚙️ Optimization (Training Objective)
5. 📏 Evaluation
6. 🚀 Deployment / Generation

Tokenization

	Word	Vocab	Embedding
Common Words	Red	Red	🟥
	Blue	Blue	🟦
Variations	Awwful	UNK	⬜
Misspell	Eastre	UNK	⬜
New Words	Vaxxed	UNK	⬜

Tokenization

Breaking down words into smaller units (subwords) that are more manageable

Cryptocurrency: Crypto + currency

Podcasting: Pod + casting

Tokenization

Breaking down words into smaller units (subwords) that are more manageable

word -> subword -> char

Tokenization

Breaking down words into smaller units (subwords) that are more manageable

I love to read books on artificial intelligence

I, love, to, read, book, ##s, on, art, ##ificial, int, ##elligence

I, , l, o, v, e, , t, o, , r, e, a, d, , b, o, o, k, s, , o, n, , a, r, t, i, f, i, c, i, a, l, , i, n, t, e, l, l, i, g, e, n, c, e,

Tokenization

Breaking down words into smaller units (subwords) that are more manageable

I love to read books on artificial intelligence

I, love, to, read, book, ##s, on, art, ##ificial, int, ##elligence

I, , l, o, v, e, , t, o, , r, e, a, d, , b, o, o, k, s, , o, n, , a, r, t, i, f, i, c, i, a, l, , i, n, t, e, l, l, i, g, e, n, c, e,

What are the advantages and disadvantages of character-level tokenization?

Byte Pair Encoding (BPE)

- Used in GPT, GPT-2, RoBERTa, BART, and DeBERTa
- Breaking down words into subwords to handle out-of-vocabulary words and improve generalization

BPE: Training

- Initialize Vocabulary
    - ASCII characters, and probably some Unicode characters and common character sequences
- Merge Frequent Pairs Iteratively
- Stop When Reaching Desired Vocabulary Size

Normalization

Removing needless whitespace, lowercasing, and/or removing accents

Héllò hôw are   ü? $\Longrightarrow$ hello how are u?

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

Pre-Tokenization

Splitting texts into small entities like words

```
pre_tokenize_str("Hello, how are  you?")
```

```
[('Hello', (0, 5)), (',', (5, 6)), ('how', (7, 10)), ('are', (11, 14)), ('you', (16, 19)), ('?', (19, 20))]
```

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

BPE: Training

Corpus

```
("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
```

$\Downarrow$

Vocabulary ["b", "g", "h", "n", "p", "s", "u"]

$\Downarrow$

```
("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
```

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

BPE: Training

Corpus

```
("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
```

⇩

Vocabulary ["b", "g", "h", "n", "p", "s", "u"]

⇩

```
("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
```

What is the most frequent pair here?

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

BPE: Training

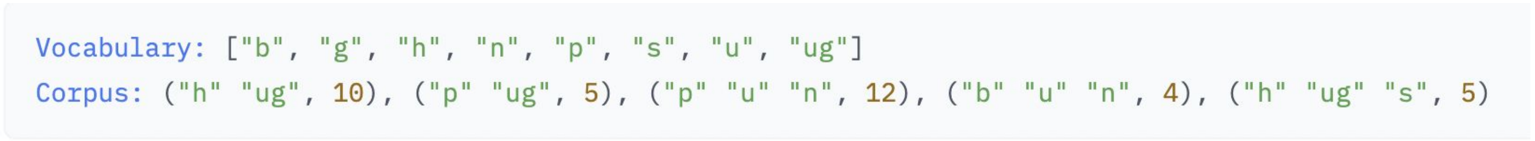
```
("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
```

$$\Downarrow$$

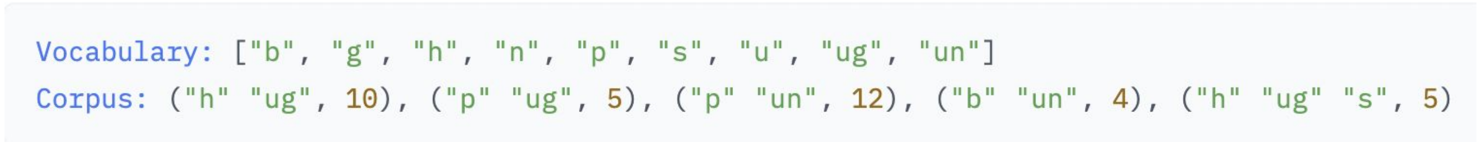
```
Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug"]
Corpus: ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)
```

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

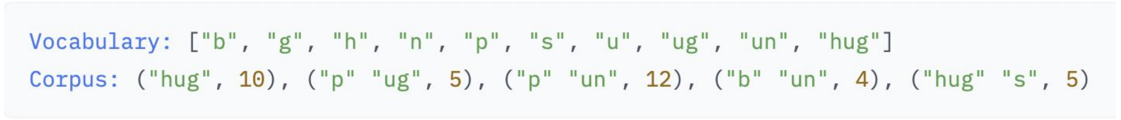
BPE: Training



```
Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug"]
Corpus: ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)
```



```
Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un"]
Corpus: ("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)
```



```
Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]
Corpus: ("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)
```

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

BPE: Tokenization

- Normalization
- Pre-tokenization
- Splitting the words into individual characters
- Applying the learned merge rules

BPE: Tokenization

Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

What are the tokenizations of “bug” and “mug” using this vocabulary?

Source: https://huggingface.co/learn/nlp-course/en/chapter6/5

WordPiece tokenization

- Used in BERT, DistilBERT, MobileBERT, Funnel Transformers, and MPNET
- Similar to BPE
    - starts from a small vocabulary including special tokens and the initial alphabet
    - Merge tokens iteratively
- But have a different merging criteria

Source: https://huggingface.co/learn/nlp-course/en/chapter6/6

WordPiece: Training

- Splitting words into characters

Hello $\Longrightarrow$ H ##e ##l ##l ##o

Discriminating characters that are in the beginning or inside a word

WordPiece: Training

- Merging rules

Prioritizing the merging of pairs where the individual parts are less frequent in the vocabulary

Score = freq(pair) / (freq(first-element)×freq(second-element))

"un", "##able"

"hu", "##gging"

WordPiece: Training

```
("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
```

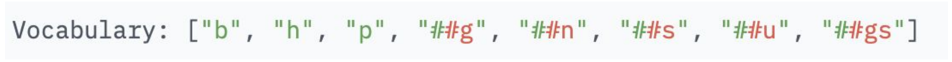
⇩

("h" "##u" "##g", 10), ("p" "##u" "##g", 5), ("p" "##u" "##n", 12), ("b" "##u" "##n", 4), ("h" "##u" "##g" "##s", 5)

Vocabulary	["b", "h", "p", "##g", "##n", "##s", "##u"]

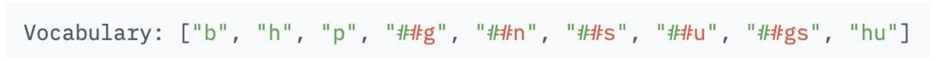
WordPiece: Training

("h" "##u" "##g", 10), ("p" "##u" "##g", 5), ("p" "##u" "##n", 12), ("b" "##u" "##n", 4), ("h" "##u" "##g" "##s", 5)



```
Vocabulary: ["b", "h", "p", "##g", "##n", "##s", "##u", "##gs"]
```

("h" "##u" "##g", 10), ("p" "##u" "##g", 5), ("p" "##u" "##n", 12), ("b" "##u" "##n", 4), ("h" "##u" "##gs", 5)



```
Vocabulary: ["b", "h", "p", "##g", "##n", "##s", "##u", "##gs", "hu"]
```

("hu" "##g", 10), ("p" "##u" "##g", 5), ("p" "##u" "##n", 12), ("b" "##u" "##n", 4), ("hu" "##gs", 5)

WordPiece: Tokenization

- Only saves the final vocabulary
- Tokenization based on the longest subword in the vocabulary

WordPiece: Tokenization

- Only saves the final vocabulary
- Tokenization based on the longest subword in the vocabulary
- If it’s not possible to find a subword in the vocabulary, the whole word is tokenized as unknown

WordPiece: Tokenization

```
Vocabulary: ["b", "h", "p", "##g", "##n", "##s", "##u", "##gs", "hu", "hug"]
```

What is the WordPiece tokenization of pun and bum using this vocabulary?

Unigram tokenization

- Used by models like AlBERT, T5, mBART, Big Bird, and XLNet

Unigram tokenization: Training

- Starts from a big vocabulary
    - E.g., large vocabulary built based on BPE, or most common substrings of words
- Remove tokens until reaching the desired vocabulary size
    - Computing a loss over the corpus given the current vocabulary
    - For each token, compute how much the loss would increase if removing the token
    - Remove the *p* percent of the tokens associated with the lowest loss increase
    - Repeat until reaching the desired size

Question

If you change the tokenizer, are you changing what the model is capable of understanding?

Questions?

What is next?

Neural Language Models: RNNs → LSTMs → Self-Attention