Lecture 3:
Language models

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center

Finite-state transducers

- FSTs define a relation between two regular languages.
- Each state transition maps (transduces) a character from the input language to a character (or a sequence of characters) in the output language.

\[ x:y \]

- By using the empty character (ε), characters can be deleted (x:ε) or inserted (ε:y).

\[ x:\epsilon \] \[ \epsilon:y \]

- FSTs can be composed (cascaded), allowing us to define intermediate representations.

Today’s lecture

How can we distinguish word salad, spelling errors and grammatical sentences?

Language models define probability distributions over the strings in a language. N-gram models are the simplest and most common kind of language model.

We'll look at how they’re defined, how to estimate (learn) them, and what their shortcomings are.

We'll also review some very basic probability theory.
Why do we need language models?

Many NLP tasks return output in natural language:

- Machine translation
- Speech recognition
- Natural language generation
- Spell-checking

Language models define probability distributions over (natural language) strings or sentences.

We can use them to score/rank possible sentences:
If \( P_{LM}(A) > P_{LM}(B) \), choose sentence \( A \) over \( B \)

Reminder:
Basic Probability Theory

Sampling with replacement

Pick a random shape, then put it back in the bag.

\[
\begin{align*}
    P(\square) &= 2/15 \\
    P(\blacksquare) &= 1/15 \\
    P(\text{blue or } \square) &= 2/15 \\
    P(\text{blue}) &= 5/15 \\
    P(\text{red}) &= 5/15 \\
    P(\text{red} | \square) &= 3/5 \\
    P(\text{blue | } \square) &= 2/5 \\
    P(\blacksquare) &= 5/15 \\
    P(\text{red} | \blacksquare) &= 3/5
\end{align*}
\]
**Sampling with replacement**

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

\[
P(\text{of}) = 3/66 \quad P(\text{her}) = 2/66 \\
P(\text{Alice}) = 2/66 \quad P(\text{sister}) = 2/66 \\
P(\text{was}) = 2/66 \quad P(.) = 4/66 \\
P(\text{to}) = 2/66 \quad P(\text{'}) = 4/66
\]

In this model, \( P(\text{English sentence}) = P(\text{word salad}) \)

---

**Probability theory: terminology**

**Trial:**
Picking a shape, predicting a word

**Sample space** \( \Omega \): The set of all possible outcomes (all shapes; all words in *Alice in Wonderland*)

**Event** \( \omega \subseteq \Omega \):
An actual outcome (a subset of \( \Omega \)) (predicting ‘the’, picking a triangle)

---

**The probability of events**

**Kolmogorov axioms:**
1) Each event has a probability between 0 and 1.
2) The null event has probability 0. The probability that any event happens is 1.
3) The probability of all disjoint events sums to 1.

\[
0 \leq P(\omega \subseteq \Omega) \leq 1 \\
P(\emptyset) = 0 \quad \text{and} \quad P(\Omega) = 1 \\
\sum_{\omega_i \subseteq \Omega} P(\omega_i) = 1 \quad \text{if} \ \forall j \neq i : \omega_i \cap \omega_j = \emptyset \\
\quad \quad \quad \text{and} \quad \bigcup_i \omega_i = \Omega
\]
Discrete probability distributions: single trials

**Bernoulli distribution** (two possible outcomes)
The probability of success (=head, yes)
The probability of head is \( p \).
The probability of tail is \( 1-p \).

**Categorical distribution** (\( N \) possible outcomes)
The probability of category/outcome \( c_i \) is \( p_i \)
\( 0 \leq p_i \leq 1 \sum p_i = 1 \)
also often (incorrectly) called Multinomial distribution

Joint and Conditional Probability

The conditional probability of \( X \) given \( Y \), \( P(X \mid Y) \),
is defined in terms of the probability of \( Y \), \( P(Y) \),
and the joint probability of \( X \) and \( Y \), \( P(X,Y) \):

\[
P(X \mid Y) = \frac{P(X,Y)}{P(Y)}
\]

\[P(\text{blue} \mid \boxed{\text{red}}) = \frac{2}{5}\]

Now, \( P(\text{English}) \geq P(\text{word salad}) \)

Conditioning on the previous word

**English**

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book' thought Alice 'without pictures or conversation?'

\[P(w_{i+1} = \text{of} \mid w_i = \text{tired}) = 1\]
\[P(w_{i+1} = \text{bank} \mid w_i = \text{the}) = \frac{1}{3}\]
\[P(w_{i+1} = \text{of} \mid w_i = \text{use}) = 1\]
\[P(w_{i+1} = \text{book} \mid w_i = \text{the}) = \frac{1}{3}\]
\[P(w_{i+1} = \text{sister} \mid w_i = \text{her}) = 1\]
\[P(w_{i+1} = \text{use} \mid w_i = \text{the}) = \frac{1}{3}\]
\[P(w_{i+1} = \text{beginning} \mid w_i = \text{was}) = \frac{1}{2}\]
\[P(w_{i+1} = \text{reading} \mid w_i = \text{was}) = \frac{1}{2}\]

**Word Salad**

beginning by very Alice but use not? reading no tired of to into sitting sister the bank and thought of without her nothing having conversations Alice a book had peeped was conversation in pictures or action in what is the use of a book 'pictures or conversation?'

\[P(w_{i+1} = \text{of} \mid w_i = \text{tired}) = 1\]
\[P(w_{i+1} = \text{bank} \mid w_i = \text{the}) = \frac{1}{3}\]
\[P(w_{i+1} = \text{of} \mid w_i = \text{use}) = 1\]
\[P(w_{i+1} = \text{book} \mid w_i = \text{the}) = \frac{1}{3}\]
\[P(w_{i+1} = \text{sister} \mid w_i = \text{her}) = 1\]
\[P(w_{i+1} = \text{use} \mid w_i = \text{the}) = \frac{1}{3}\]
\[P(w_{i+1} = \text{beginning} \mid w_i = \text{was}) = \frac{1}{2}\]
\[P(w_{i+1} = \text{reading} \mid w_i = \text{was}) = \frac{1}{2}\]
The chain rule

The joint probability $P(X,Y)$ can also be expressed in terms of the conditional probability $P(X | Y)$

$$P(X, Y) = P(X | Y) P(Y)$$

This leads to the so-called chain rule:

$$P(X_1, X_2, \ldots, X_n) = P(X_1) P(X_2 | X_1) P(X_3 | X_2, X_1) \cdots P(X_n | X_1, \ldots X_{n-1})$$

$$= P(X_1) \prod_{i=2}^{n} P(X_i | X_1 \ldots X_{i-1})$$

Independence

Two random variables $X$ and $Y$ are independent if

$$P(X, Y) = P(X) P(Y)$$

If $X$ and $Y$ are independent, then $P(X | Y) = P(X)$:

$$P(X | Y) = \frac{P(X, Y)}{P(Y)}$$

$$= \frac{P(X) P(Y)}{P(Y)}$$

$$= P(X)$$

Probability models

Building a probability model consists of two steps:

1. Defining the model
2. Estimating the model’s parameters
   (= training/learning )

Models (almost) always make independence assumptions.

That is, even though $X$ and $Y$ are not actually independent, our model may treat them as independent.

This reduces the number of model parameters that we need to estimate (e.g. from $n^2$ to $2n$)

Language modeling with n-grams
A language model over a vocabulary \( V \) assigns probabilities to strings drawn from \( V^* \).

Recall the chain rule:

\[
P(w_1 \ldots w_i) = P(w_1) P(w_2 | w_1) P(w_3 | w_1 w_2) \ldots P(w_i | w_1 \ldots w_{i-1})
\]

An n-gram language model assumes each word depends only on the last \( n-1 \) words:

\[
P_{\text{n-gram}}(w_1 \ldots w_i) := P(w_1) P(w_2 | w_1) \ldots P(w_i | w_{i-n+1} \ldots w_{i-1})
\]

Language modeling with N-grams

N-gram models assume each word (event) depends only on the previous \( n-1 \) words (events).

Such independence assumptions are called Markov assumptions (of order \( n-1 \)).

\[
P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1})
\]

Estimating N-gram models

1. Bracket each sentence by special start and end symbols:

   \(<s>Alice was beginning to get very tired... </s>
   (We only assign probabilities to strings \(<s>...</s>)

2. Count the frequency of each n-gram....

   \( C(<s>Alice) = 1, C(Alice was) = 1, \ldots \)

3. .... and normalize these frequencies to get the probability:

   \[
P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}
\]

   This is called a relative frequency estimate of \( P(w_n | w_{n-1}) \)

Start and End symbols \(<s>... </s>

Why do we need a start-of-sentence symbol?
This is just a mathematical convenience, since it allows us to write e.g. \( P(w_1 | <s>) \) for the probability of the first word in analogy to \( P(w_{i+1} | w_i ) \) for any other word.

Why do we need an end-of-sentence symbol?
This is necessary if we want to compare the probability of strings of different lengths (and actually define a probability distribution over \( V^* \)).

We include \(<s>\) in the vocabulary \( V \), require that each string ends in \(<s>\) and that \(<s>\) can only appear at the end of sentences, and estimate \( P(w_{i+1} = <s> | w_i ) \).
Parameter estimation (training)

Parameters: the actual probabilities
\[ P(w_i = \text{`the' } | \ w_{i-1} = \text{`on' }) = ??? \]

We need (a large amount of) text as training data to estimate the parameters of a language model.

The most basic estimation technique: relative frequency estimation (= counts)
\[ P(w_i = \text{`the' } | \ w_{i-1} = \text{`on' }) = C(\text{`on the'}) / C(\text{`on'}) \]
Also called Maximum Likelihood Estimation (MLE)

MLE assigns all probability mass to events that occur in the training corpus.

How do we use language models?

Independently of any application, we can use a language model as a random sentence generator (i.e. we sample sentences according to their language model probability)

Systems for applications such as machine translation, speech recognition, spell-checking, generation, often produce multiple candidate sentences as output.
- We prefer output sentences \( S_{Out} \) that have a higher probability
- We can use a language model \( P(S_{Out}) \) to score and rank these different candidate output sentences, e.g. as follows:
\[ \text{argmax}_{S_{Out}} P(S_{Out} | \text{Input}) = \text{argmax}_{S_{Out}} P(\text{Input} | S_{Out})P(S_{Out}) \]

Generating from a distribution

How do you generate text from an \( n \)-gram model?

That is, how do you sample from a distribution \( P(X \mid Y=y) \)?
- Assume \( X \) has \( N \) possible outcomes (values): \( \{x_1, \ldots, x_N\} \) and \( P(X=x_i \mid Y=y) = p_i \)
- Divide the interval \([0,1]\) into \( N \) smaller intervals according to the probabilities of the outcomes
- Generate a random number \( r \) between 0 and 1.
- Return the \( x_i \) whose interval the number is in.

Using n-gram models to generate language
Generating the Wall Street Journal

unigram: Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives

bigram: 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

trigram: 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

Intrinsic vs Extrinsic Evaluation

How do we know whether one language model is better than another?

There are two ways to evaluate models:
- **intrinsic evaluation** captures how well the model captures what it is supposed to capture (e.g. probabilities)
- **extrinsic (task-based) evaluation** captures how useful the model is in a particular task.

Both cases require an **evaluation metric** that allows us to measure and compare the performance of different models.

How do we evaluate models?

Define an **evaluation metric (scoring function)**.

We will want to measure how similar the predictions of the model are to real text.

Train the model on a ‘seen’ training set

Perhaps: tune some parameters based on **held-out data** (disjoint from the training data, meant to emulate unseen data)

Test the model on an **unseen test set**

(usually from the same source (e.g. WSJ) as the training data) Test data must be disjoint from training and held-out data Compare models by their scores (more on this next week).
Intrinsic Evaluation of Language Models: Perplexity

Perplexity

The inverse of the probability of the test set, normalized by the number of tokens in the test set.

Assume the test corpus has N tokens, $w_1 \ldots w_N$

If the LM assigns probability $P(w_1, \ldots, w_{i-n})$ to the test corpus, its perplexity, $PP(w_1 \ldots w_N)$, is defined as:

$$PP(w_1 \ldots w_N) = P(w_1 \ldots w_N)^{-\frac{1}{n}}$$

$$= \sqrt[n]{P(w_1 \ldots w_N)}$$

A LM with lower perplexity is better because it assigns a higher probability to the unseen test corpus.

Perplexity $PP(w_1 \ldots w_n)$

Given a test corpus with N tokens, $w_1 \ldots w_N$, and an n-gram model $P(w_i | w_{i-1}, \ldots, w_{i-n+1})$ we compute its perplexity $PP(w_1 \ldots w_N)$ as follows:

$$PP(w_1 \ldots w_N) = P(w_1 \ldots w_N)^{-\frac{1}{n}}$$

$$= \sqrt[n]{P(w_1 \ldots w_N)}$$

$$= \sqrt[n]{\prod_{i=1}^{n} P(w_i | w_{i-1} \ldots w_{i-n+1})}$$

(Chain rule)

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i | w_{i-1} \ldots w_{i-n+1})}}$$

(N-gram model)
Practical issues

Since language model probabilities are very small, multiplying them together often yields to underflow.

It is often better to use logarithms instead, so replace

\[
PP(w_1...w_N) = \prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1},...,w_{i-n+1})}
\]

with

\[
PP(w_1...w_N) = \exp \left( -\frac{1}{N} \sum_{i=1}^{N} \log P(w_i|w_{i-1},...,w_{i-n+1}) \right)
\]

Perplexity and LM order

Bigram LMs have lower perplexity than unigram LMs
Trigram LMs have lower perplexity than bigram LMs ...

Example from the textbook (WSJ corpus)

<table>
<thead>
<tr>
<th>Perplexity</th>
<th>Unigram</th>
<th>Bigram</th>
<th>Trigram</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>962</td>
<td>170</td>
<td>109</td>
</tr>
</tbody>
</table>

Intrinsic vs. Extrinsic Evaluation

Perplexity tells us which LM assigns a higher probability to unseen text

This doesn’t necessarily tell us which LM is better for our task (i.e. is better at scoring candidate sentences)

Task-based evaluation:
- Train model A, plug it into your system for performing task T
- Evaluate performance of system A on task T.
- Train model B, plug it in, evaluate system B on same task T.
- Compare scores of system A and system B on task T.
Word Error Rate (WER)

Originally developed for speech recognition.

How much does the predicted sequence of words differ from the actual sequence of words in the correct transcript?

\[
WER = \frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{\text{Actual words in transcript}}
\]

Insertions: “eat lunch” → “eat a lunch”
Deletions: “see a movie” → “see movie”
Substitutions: “drink ice tea” → “drink nice tea”

But….

… unseen test data will contain unseen words

Generating Shakespeare

<table>
<thead>
<tr>
<th>Interrogative</th>
<th>Colloquial</th>
</tr>
</thead>
<tbody>
<tr>
<td>To him swallowed confess hear both. Whish. Of save on tray for are ay device and rote lite have</td>
<td>What means, sir? I confess she? then all sorts, he is trim, captain.</td>
</tr>
<tr>
<td>Every enter now severally so, let</td>
<td>Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry.</td>
</tr>
<tr>
<td>Hill he late speaks; or! a more to leg less first you enter</td>
<td>Are where exerent and sighs have rise excellency took of. Sleep knave we. near; vile like</td>
</tr>
<tr>
<td>Are where exerent and sighs have rise excellency took of. Sleep knave we. near; vile like</td>
<td>Enter Menehun, if it so many good direction found’st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exclamatory</th>
<th>Exclamatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>This shall forbid it should be branded, if renown made it empty.</td>
<td>Sweet prince, Falstaff shall die. Harry of Monmouth’s grave.</td>
</tr>
<tr>
<td>Indeed the duke; and had a very good friend.</td>
<td>This shall forbid it should be branded, if renown made it empty.</td>
</tr>
<tr>
<td>Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.</td>
<td>Indeed the duke; and had a very good friend.</td>
</tr>
<tr>
<td>But…</td>
<td>Indeed the short and the long. Marry, 'tis a noble Lepidus.</td>
</tr>
</tbody>
</table>

But…. 

… unseen test data will contain unseen words

Getting back to Shakespeare…
Shakespeare as corpus

The Shakespeare corpus consists of $N=884,647$ word tokens and a vocabulary of $V=29,066$ word types.

Shakespeare produced 300,000 bigram types out of $V^2=844$ million possible bigram types.

99.96% of possible bigrams don’t occur in the corpus.

Our relative frequency estimate assigns non-zero probability to only 0.04% of the possible bigrams. That percentage is even lower for trigrams, 4-grams, etc. 4-grams look like Shakespeare because they are Shakespeare!

MLE doesn’t capture unseen events

We estimated a model on 440K word tokens, but:

Only 30,000 word types occur in the training data
Any word that does not occur in the training data has zero probability!

Only 0.04% of all possible bigrams (over 30K word types) occur in the training data
Any bigram that does not occur in the training data has zero probability (even if we have seen both words in the bigram)

So….

… we can’t actually evaluate our MLE models on unseen test data (or system output)…

… because both are likely to contain words/n-grams that these models assign zero probability to.

We need language models that assign some probability mass to unseen words and n-grams.

We will get back to this on Friday.

Zipf’s law: the long tail

In natural language:
- A small number of events (e.g. words) occur with high frequency
- A large number of events occur with very low frequency
To recap....

Today’s key concepts

N-gram language models
Independence assumptions
Relative frequency (maximum likelihood) estimation
Evaluating language models: Perplexity, WER
Zipf’s law

Today’s reading:
Jurafsky and Martin, Chapter 4, sections 1-4

Friday’s lecture: Handling unseen events!