NLP1

Language Modelling

Wilker Aziz
probabll.github.io
w.aziz@uva.nl

ILLC

Where are we at?

HC1a

- What makes NLP hard
- Text classification

HC1b (today)

- **Language modelling**

Goals for this class

- what LMs **are**
- how to **design** LMs
- how to **estimate** LMs
- how to **evaluate** LMs

Intuition

On Mentimeter...

Table of contents

Language Model

Language Model

What LMs are

Language model

A language model is a **probability distribution** over the set of all strings in a language.

Being a probability distribution means that

1. an LM can assign probability to text
2. you can generate text by drawing samples from the LM

Applications

- Order alternative sentences (e.g., in speech recognition)
- Generate text in context (e.g., autocomplete)
- Backbone of various NLP systems: translation, summarisation, chatbots

Let's try to assign probability ourselves

On Mentimeter...

Probabilities are Made-Up Quantities

Probability IS NOT a property of text,[1]

- we **cannot** learn LMs by regressing from text to "observed target probabilities"

Probability is an expression of preference (by a model or observer),[2]

- we **can** learn to assign probability to text by
  - designing a statistical model of how texts come about
  - and optimising the parameters of this model using observed data and a statistical criterion of our choice.

[1] De Finetti's seminal *Theory of Probability* starts with a provocative claim: PROBABILITY DOES NOT EXIST.

[2] On occasion, we can make it capture *sample frequency* in a population.

Let's see if we are on the same page

On Mentimeter...

A Good LM is...

A probability distribution whose samples *resemble* observed text.[3]

Examples:

- if the typical sentence has 30 words, sampling from a good LM will reproduce that pattern;
- if the typical sentence has SUBJ VERB OBJ (in this order), samples from a good LM will exhibit that pattern too;
- etc.

[3] This is a statistical notion of 'goodness', we will discuss other notions later.

Language Model

How to design LMs

Designing LMs

Since an LM is a probability distribution, designing one requires choosing:

- A **sample space.** The set of outcomes that the LM can generate and assign probability to.
- A **probability mass function (pmf).** A function mapping each and every outcome in the sample space to its assigned probability mass.

Text as a Data Type

We regard text as a finite sequence of discrete symbols which we generally refer to as ‘words’ (or, even more generally, as ‘tokens’).

Digital text is a sequence of characters. Using a *tokenisation algorithm* we ‘segment’ digital text into a sequence of tokens.

Example: with a tokenisation procedure based on spaces and punctuation, the English text what a nice dog! is regarded as a sequence of 5 tokens: (what, a, nice, dog, !).[4]

[4] Modern, general-purpose tokenisers are based on compression algorithms [Sennrich et al., 2016].

Sample Space

Start with a vocabulary of ‘words’ $\mathcal{W}$ (for example, all unique tokens found in some large, representative dataset).

The sample space of choice is typically the set of all finite-length sequences made of symbols from this vocabulary. This set is what we call the ‘language’ (in a formal, non-linguistic sense), it is denoted by $\mathcal{W}^*$.

Example

- with $\mathcal{W} = \{\text{a}, \text{cat}, \text{dog}, \text{nice}\}$
- then a nice cat and a dog are sentences in the language $\mathcal{W}^*$, just like a a or nice a nice a nice, but a cute dog isn’t.

Formalisation

W is a random word. An outcome w is a symbol in a vocabulary $\mathcal{W}$ of size V.

$X = \langle W_1, \ldots, W_L \rangle$ is a random *sequence* of L words. We can also denote it $W_{1:L}$. An outcome $\langle w_1, \ldots, w_l \rangle$ is a sequence in $\mathcal{W}^*$, that is, a sequence of l symbols from $\mathcal{W}$.

A language model is a mechanism to assign a probability value $P_X(w_{1:l})$ to **each and every** outcome $w_{1:l} \in \mathcal{W}^*$.

We now design a pmf to map $w_{1:l}$ to its probability mass.

Challenge

P_X is a distribution over a countably infinite space of variable-length sequences.

Intuition. To see that this is a real challenge, let's list outcomes of X in a table and associate each outcome with a scalar *parameter* for the outcome's probability mass.

Unique id	Outcome x	Probability $P_X(x)$
1	nice!	θ_1
2	a cat!	θ_2
3	a cute cat!	θ_3
4	a nasty cat!	θ_4
5	what a cute cat!	θ_5
6	what a nasty cat!	θ_6
	...	

How many probabilities do we need to store?

Tabular Representation of Discrete Distributions

What we just tried to use is a *tabular representation* of the pmf of a discrete random variable.

If it were a viable representation, we would assing probability to $X = x$ via a table lookup:

$$P_X(x) = \theta_{\mathsf{id}(x)}$$

The tabular representation is based on enumeration of outcomes and its parameters are statistically independent of one another. This is inefficient (computationally and statistically), and, for distributions with countably infinite sample spaces, this is unusable.

Factorisation

Key Idea

Think of an outcome $w_{1:l} \in \mathcal{W}^*$ as a *structure* (it clearly is). Imagine a procedure by which you could *derive* this structure in a sequence of *steps*.

We can then re-express the probability of any one sequence $w_{1:l}$ in $\mathcal{W}^*$ using probabilities assigned to the steps that jointly derive it.

If we design this well, we will be able to work with distributions whose sample spaces are more manageable.

Example – Deriving Text

Consider the sequence $x = \langle \text{He, went, to, the, store, EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle \rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.

[5]EOS marks the end of the sequence, the need for it will become clear.

Example – Deriving Text

Consider the sequence $x = \langle \text{He, went, to, the, store, EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle \rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.
2. Given h_2, choose the second word: went. Make $h_3 \leftarrow h_2 \circ \langle \text{went} \rangle$.

[5] EOS marks the end of the sequence, the need for it will become clear.

Example – Deriving Text

Consider the sequence $x = \langle \text{He}, \text{went}, \text{to}, \text{the}, \text{store}, \text{EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle \rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.
2. Given h_2, choose the second word: went. Make $h_3 \leftarrow h_2 \circ \langle \text{went} \rangle$.
3. Given h_3, choose the third word: to. Make $h_4 \leftarrow h_3 \circ \langle \text{to} \rangle$.

[5] EOS marks the end of the sequence, the need for it will become clear.

Example – Deriving Text

Consider the sequence $x = \langle \text{He, went, to, the, store, EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle\rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.
2. Given h_2, choose the second word: went. Make $h_3 \leftarrow h_2 \circ \langle \text{went} \rangle$.
3. Given h_3, choose the third word: to. Make $h_4 \leftarrow h_3 \circ \langle \text{to} \rangle$.
4. Given h_4, choose the fourth word: the. Make $h_5 \leftarrow h_4 \circ \langle \text{the} \rangle$.

[5] EOS marks the end of the sequence, the need for it will become clear.

Example – Deriving Text

Consider the sequence $x = \langle \text{He, went, to, the, store, EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle \rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.
2. Given h_2, choose the second word: went. Make $h_3 \leftarrow h_2 \circ \langle \text{went} \rangle$.
3. Given h_3, choose the third word: to. Make $h_4 \leftarrow h_3 \circ \langle \text{to} \rangle$.
4. Given h_4, choose the fourth word: the. Make $h_5 \leftarrow h_4 \circ \langle \text{the} \rangle$.
5. Given h_5, choose the fifth word: store. Make $h_6 \leftarrow h_5 \circ \langle \text{store} \rangle$.

[5] EOS marks the end of the sequence, the need for it will become clear.

Example – Deriving Text

Consider the sequence $x = \langle \text{He, went, to, the, store, EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle\rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.
2. Given h_2, choose the second word: went. Make $h_3 \leftarrow h_2 \circ \langle \text{went} \rangle$.
3. Given h_3, choose the third word: to. Make $h_4 \leftarrow h_3 \circ \langle \text{to} \rangle$.
4. Given h_4, choose the fourth word: the. Make $h_5 \leftarrow h_4 \circ \langle \text{the} \rangle$.
5. Given h_5, choose the fifth word: store. Make $h_6 \leftarrow h_5 \circ \langle \text{store} \rangle$.
6. Given h_6, choose the sixth word: EOS. Make $x = h_6 \circ \langle \text{EOS} \rangle$.

[5] EOS marks the end of the sequence, the need for it will become clear.

Example – Deriving Text

Consider the sequence $x = \langle \text{He, went, to, the, store, EOS} \rangle$.[5] Now think of x as the result of incrementally expanding an empty sequence, one symbol at a time. Start with $h_1 = \langle \rangle$, then

1. Given h_1, choose the first word: He. Make $h_2 \leftarrow h_1 \circ \langle \text{He} \rangle$.
2. Given h_2, choose the second word: went. Make $h_3 \leftarrow h_2 \circ \langle \text{went} \rangle$.
3. Given h_3, choose the third word: to. Make $h_4 \leftarrow h_3 \circ \langle \text{to} \rangle$.
4. Given h_4, choose the fourth word: the. Make $h_5 \leftarrow h_4 \circ \langle \text{the} \rangle$.
5. Given h_5, choose the fifth word: store. Make $h_6 \leftarrow h_5 \circ \langle \text{store} \rangle$.
6. Given h_6, choose the sixth word: EOS. Make $x = h_6 \circ \langle \text{EOS} \rangle$.

We can now assign probability to $X = x$ by assigning probability to each of these ‘steps’, all of which must be taken in order to reproduce x.

[5] EOS marks the end of the sequence, the need for it will become clear.

Example – Assigning Probability

$$
\begin{aligned}
&P_X(\langle \text{He, went, to, the, store} \rangle) \\
&\triangleq P_{W|H}(\text{He}|\langle\rangle) \\
&\quad \times P_{W|H}(\text{went}|\langle \text{He} \rangle) \\
&\quad \times P_{W|H}(\text{to}|\langle \text{He}, \text{went} \rangle) \\
&\quad \times P_{W|H}(\text{the}|\langle \text{He}, \text{went}, \text{to} \rangle) \\
&\quad \times P_{W|H}(\text{store}|\langle \text{He}, \text{went}, \text{to}, \text{the} \rangle) \\
&\quad \times P_{W|H}(\text{EOS}|\langle \text{He}, \text{went}, \text{to}, \text{the}, \text{store} \rangle)
\end{aligned}
$$

We are specifying how the distribution P_X assigns probability to the text $\langle$He, went, to, the, store$\rangle$. We choose to compute that number by multiplying probabilities that some other distributions assign to the words in the text as we traverse the sequence from left-to-right. Each time, we assign probability to a word, we do it *conditioned on* an ordered **history** of words that precede it.

Designing LMs via Chain Rule of Probabilities

Our most general LM assigns probability to $w_{1:l}$ as defined below:

$$P_X(w_{1:l}) \triangleq \prod_{i=1}^{l} P_{W|H}(w_i|w_{<i}) \tag{1}$$

This is what we call an *autoregressive factorisation* of the probability of a sequence.

We will work on the design of the conditional distributions that appear on the right-hand side of the equation.

History-Conditioned Word Distributions

For any given history h, such as $\langle \text{He}, \text{went}, \text{to}, \text{the} \rangle$, we need to be able to assign probability $P_{W|H}(w|h)$ to any symbol w in the vocabulary—there are only $V = |\mathcal{W}|$ such symbols!

In tabular representation, the pmf of the random variable $W|H = h$, for any one history h, can be represented by a unit-norm, V-dimensional vector $\boldsymbol{\theta}^{(h)}$ of probability masses.

id	w	$P_{W\|H}(w\|h)$
1	a	$\theta_1^{(h)}$
2	amazing	$\theta_2^{(h)}$
	...	
V	zyzzyva	$\theta_V^{(h)}$

Tabular representation

$P_{W|H=h}$ is what we call a *conditional probability distribution* (cpd).

Assigning Probability or Generating Outcomes

Our procedure to assign probability to an outcome also prescribes a *sampler* (or *simulator*, or *generator*), that is, an algorithm to *generate* outcomes from the LM distribution P_X.

This procedure is also known as a **generative story**.

Generative Story

1. Start with an empty history $h_1 = \langle\rangle$. Set $i = 1$.
2. Condition on the available history h_i and draw a word w_i with probability $P_{W|H}(w_i|h_i)$ extending the history with it.
3. If w_i is a special end-of-sequence symbol (EOS), terminate, else increment i and repeat (2).

This specifies a **factorisation** of P_X in terms of elementary factors of the kind $P_{W|H}$.

Summary

- An LM is a distribution P_X over the space of all texts
- Rather than working with P_X directly, we re-express it via chain rule
- For any given history h, we must be able to prescribe a distribution $P_{W|H=h}$ over the vocabulary
- The vocabulary is finite, so the pmf of $P_{W|H=h}$ is representable by a tractable V-dimensional vector.
- There's no limit to the set of possible histories (any sequence of any number of words, so long as it does not end in EOS).

We deal with this next!

Parameterisation

Factorising and Parameterising

Factorising $P_X(x)$ means decomposing this quantity using a product of 'simpler' factors.[6] We have used chain rule to decompose it as $P_X(w_{1:l}) = \prod_{i=1}^{l} P_{W|H}(w|h)$.

We now design a mechanism to compute $P_{W|H}(w|h)$ for any choice of (h, w). This design is what we call a *parameterisation*.

[6] Little analogy: 24 can be factorised into a product of prime numbers: $2 \times 2 \times 2 \times 3 = 2^3 \times 3$.

Major Ideas in NLP

Assign probability to any $w \in \mathcal{W}$ given any $h \in \mathcal{W}^*$

1. using the *relative frequency* of $h \circ \langle w \rangle$, as observed in a large corpus;

These are all Frequentist ideas. For Bayesian ideas, see for example Cohen and Hirst [2019].

Major Ideas in NLP

Assign probability to any $w \in \mathcal{W}$ given any $h \in \mathcal{W}^*$

1. using the *relative frequency* of $h \circ \langle w \rangle$, as observed in a large corpus;
2. informed by the *count* of $h \circ \langle w \rangle$, and of its subsequences, in a large corpus;

These are all Frequentist ideas. For Bayesian ideas, see for example Cohen and Hirst [2019].

Major Ideas in NLP

Assign probability to any $w \in \mathcal{W}$ given any $h \in \mathcal{W}^*$

1. using the *relative frequency* of $h \circ \langle w \rangle$, as observed in a large corpus;
2. informed by the *count* of $h \circ \langle w \rangle$, and of its subsequences, in a large corpus;
3. using a *log-linear model* with features $\phi(h) \in \mathbb{R}^D$;

These are all Frequentist ideas. For Bayesian ideas, see for example Cohen and Hirst [2019].

Major Ideas in NLP

Assign probability to any $w \in \mathcal{W}$ given any $h \in \mathcal{W}^*$

1. using the *relative frequency* of $h \circ \langle w \rangle$, as observed in a large corpus;
2. informed by the *count* of $h \circ \langle w \rangle$, and of its subsequences, in a large corpus;
3. using a *log-linear model* with features $\phi(h) \in \mathbb{R}^D$;
4. using a *non-linear model* to map from h directly to the (parameters of the) pmf.

We discuss 1 and 2 today, 3 and 4 later in the course.

These are all Frequentist ideas. For Bayesian ideas, see for example Cohen and Hirst [2019].

Relative Frequency

If we want to assign probability to say store given say he went to the, we use the frequency of **he went to the store** relative to the frequency of he went to the • (that is followed by *any* known word) in a large corpus. Or, generically, for a word w and a history h:

$$P_{W|H}(w|h) = \frac{\text{count}_{HW}(h, w)}{\sum_{o \in \mathcal{W}} \text{count}_{HW}(h, o)} \tag{2}$$

This corresponds to maximum likelihood estimation (MLE) for tabular Categorical cpds of the kind $P_{W|H=h}$.

Relative Frequency

If we want to assign probability to say store given say he went to the, we use the frequency of **he went to the store** relative to the frequency of he went to the • (that is followed by *any* known word) in a large corpus. Or, generically, for a word w and a history h:

$$P_{W|H}(w|h) = \frac{\text{count}_{HW}(h, w)}{\sum_{o \in \mathcal{W}} \text{count}_{HW}(h, o)} \tag{2}$$

This corresponds to maximum likelihood estimation (MLE) for tabular Categorical cpds of the kind $P_{W|H=h}$.

Do you see any problem with this idea?

Data Sparsity!

If we have not seen a given h followed by a certain w, the relative frequency is 0. If we have not seen a given h, the relative frequency is not even defined.

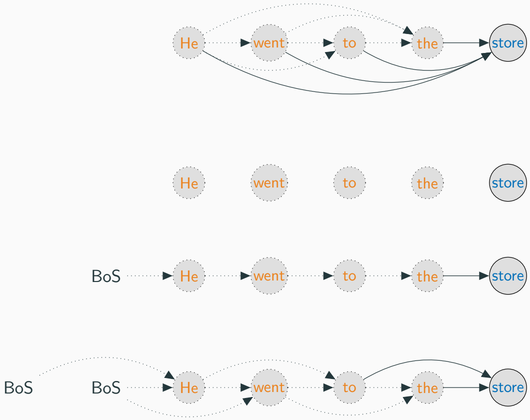
Unavoidable truth about empirical methods: not seeing something is not evidence of it not being possible. It's just data sparsity speaking.

An answer that was popular for decades: change the *factorisation*, simplifying h to retain only words that are closest to the next one.

NGram LMs

Make a simplifying **Markov assumption**:

Next word is *conditionally independent* of all but the $N - 1$ preceding words.



Top-down: autoregressive LM, unigram LM (N=1), bigram LM (N=2), trigram LM (N=3).

Conditional Independence and Markov Assumptions

The key idea in the NGram LM is to make a conditional independence assumption:

$$P_X(w_{1:l}) \overset{\text{ind.}}{=} \prod_{i=1}^{l} P_{W|H}(w_i | \langle w_{i-N+1}, \ldots, w_{i-1} \rangle) \tag{3}$$

a word is independent on all but the recent history of $N-1$ words. This is the so-called **Markov assumption** of order $N-1$.

Tabular CPDs for NGram LMs

Store relative frequencies of observed NGrams in a table.

If we design a 3-gram LM (i.e., $N = 3$) to assign probability to say store given say $\langle$he, went, to, the$\rangle$, we first truncate the history to the last 2 words $\langle$to, the$\rangle$ such that it has length 2 and then use the frequency of **to the store** relative to the frequency of to the • (that is followed by *any* known word) in a large corpus.

Generically, for a word w, a history h and NGram size N:

$$P_{W|H}(w|h) = \frac{\text{count}_{HW}(\text{last}_{N-1}(h), w)}{\sum_{o \in \mathcal{W}} \text{count}_{HW}(\text{last}_{N-1}(h), o)} \tag{4}$$

Tabular CPDs for NGram LMs

Store relative frequencies of observed NGrams in a table.

If we design a 3-gram LM (i.e., $N = 3$) to assign probability to say store given say $\langle$he, went, to, the$\rangle$, we first truncate the history to the last 2 words $\langle$to, the$\rangle$ such that it has length 2 and then use the frequency of **to the store** relative to the frequency of to the • (that is followed by *any* known word) in a large corpus.

Generically, for a word w, a history h and NGram size N:

$$P_{W|H}(w|h) = \frac{\text{count}_{HW}(\text{last}_{N-1}(h), w)}{\sum_{o \in \mathcal{W}} \text{count}_{HW}(\text{last}_{N-1}(h), o)} \tag{4}$$

Any problems with this?

Smoothing

Reserve probability for unseen NGrams:

$$
\begin{aligned}
P_{W|H}(w|h) &= \frac{\mathrm{count}_{HW}(h,w) + \alpha(h)}{\sum_{o \in \mathcal{W}} (\mathrm{count}_{HW}(h,o) + \alpha(h))} \\
&= \frac{\mathrm{count}_{HW}(h,w) + \alpha(h)}{V \times \alpha(h) + \mathrm{count}_{H}(h)}
\end{aligned} \tag{5}
$$

Example with $\alpha(h) = 1$ a.k.a. 'Laplace Smoothing'

- $\mathrm{count}_H(\langle \text{a}, \text{nice} \rangle) = 100$
- rabbit $\in \mathcal{W}$
- but $\mathrm{count}_{HW}(\langle \text{a}, \text{nice} \rangle, \text{rabbit}) = 0$.

Then, $P_{W|H}(\text{rabbit}|\langle \text{a}, \text{nice} \rangle) = \frac{1}{V+100}$.

Tip. When implementing smoothed models, it's easier to store *counts* (rather than parameters), because counts are sparse (many 0s) but parameters aren't.

Unknown Words

Here the situation is a little different. We want to deal with a symbol that's not at all in the vocabulary.

Idea: augment the vocabulary with a placeholder symbol such as UNK, whenever you encounter an unknown symbol in the future (e.g., "hare") treat it as UNK.

In combination with smoothing, this should help avoid assigning 0 probabilities.

Exercises

On Mentimeter...

Memorise phrases

The NGram LM makes up a sentence by gluing phrases, which it memorises in its tabular cpds along with their counts.

An increase in the order has an exponential cost: $V^N \to V^{N+1}$

The longer the history, the less likely it is that we have seen it.

Most of the possible history-word pairs will never be seen.

Tricks: smoothing, interpolation, backoff, etc. For an overview (though I consider it *optional knowledge*) see section 3.5 of textbook.

Limitations

Our Markov assumptions are motivated by convenience alone, long range dependency is a very common thing in natural languages.

To overcome this we need to move beyond ‘storing’ probabilities (or the counts that are used to compute them). The key idea that will unlock the most powerful LMs is to learn to *predict* probability masses using parametric (log-linear or nonlinear) models.

Evaluation

Intrinsic

We assess the average surprisal (negative log probability) that our model assigns to heldout texts $\{x^{(1)}, \ldots, x^{(S)}\}$:

$$\frac{1}{S} \sum_{s=1}^{S} \log P_X(x^{(s)}) \tag{6}$$

For ease of interpretation, we re-express it in terms of **perplexity per token**, a measure of average confusion.[10]

Required reading: section 3.8 of textbook.

[10] If perplexity per token is 5, on average across histories, the model's uncertainty over the next token spreads over 5 candidates from the vocabulary.

Extrinsic

Plug the LM in a task (e.g., autocomplete) and measure the performance in that task.

Statistical

Compare how the statistics of *generated* text distribute in relation to statistics of *observed* text.

Examples:

- Meister and Cotterell [2021]
- Giulianelli et al. [2023]

Be Critical

Your statistical model is as good as your statistical assumptions, your estimation procedure, and the data you use to fit it.

Most assumptions are wrong or insufficient. Any dataset (however large) is at best a snippet of language production by some groups of speakers: not good enough to represent a whole world of speakers, not good enough to represent any one specific group of speakers.

Models are not trained to *understand*, they are not trained to *respond*, they are not trained to *comply with the values of the humans using it*, they are not trained to *produce factually correct text*, they are trained such that their samples **look like** they could have been found in the training data.

Summary

- LMs are distributions over texts
- Chain rule is the key to prescribing an LM
- Classic NGram LMs: Markov assumption + tabular parameterisation
- Tabular parameterisation is statistically inefficient
- Modern approaches parameterise cpds using NNs

Check our website for some auxiliary self-study material.

References

Shay Cohen and Graeme Hirst. *Bayesian Analysis in Natural Language Processing, Second Edition*. Morgan & Claypool Publishers, 2nd edition, 2019. ISBN 1681735261.

Mario Giulianelli, Joris Baan, Wilker Aziz, Raquel Fernández, and Barbara Plank. What comes next? evaluating uncertainty in neural text generators against human production variability. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14349–14371, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.887. URL `https://aclanthology.org/2023.emnlp-main.887`.

Clara Meister and Ryan Cotterell. Language model evaluation beyond perplexity. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5328–5339, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.414. URL `https://aclanthology.org/2021.acl-long.414`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:

10.18653/v1/P16-1162. URL
https://aclanthology.org/P16-1162.