Language models

Jelke Bloem & Giovanni Colavizza

Text Mining
Amsterdam University College

February 19, 2024

Overview

1. Language modelling
2. Estimating n-gram probabilities
3. Evaluating language models
4. Research application: Text complexity
5. Elementary probability theory

Language modelling

Language modelling

- (Statistical) **Language Models** assign probabilities to sequences of words from a vocabulary.
- **Goal**: to compute the probability of a sequence of words:
  - $P(w) = P(w_1, w_2, .., w_n)$
  - P("Call me Ishmael") = 0.0001
- Also used to estimate the probability of a new word:
  - $P(w_n | w_1, w_2, .., w_{n-1})$
- Note that the set of word sequences of any length from any vocabulary is infinite. Practically, well-formed sequences for a given language are much fewer (**plausible sequences**).
- Language models have very many applications and often underpin other techniques (e.g., translation, question answering, language understanding, language generation, etc.).

N-gram language modelling

The simplest language models.

- **n-gram**: a sequence of n words.
  - **unigram**: every word is assumed to be completely independent.
  - **bigram**: a sequence of 2 words. E.g., "your homework", "the house", "my bed" ..
  - **trigram**: a sequence of 3 words. E.g., "your best friend", "my worst nightmare", ..
- Clearly, some combinatorial restrictions of word sequences are determined by the grammar.
  - E.g., a verb has a subject; an article is used with a name, ..
  - The probability of a word w_i in a sequence w is not independent from its **context**.

Identifying bigrams

$\langle s \rangle$ *I am paid by the word so I always write the shortest words possible* $\langle /s \rangle$

- number of tokens $n = 14$ (or 15, or 16)
- number of bigrams 13 (or 14, or 15).
- Important: the **end symbol (EoS)** is used to model the fact that a word is at the end of a sentence. EoS is needed to make the model a true probability distribution over all sequence lengths.

Identifying bigrams with NLTK

```
>>> import nltk
>>> sentence = ["I", "am", "paid", "by", "the", "word",
"so", "I", "always", "write", "the", "shortest", "words",
"possible"]
```

Definition of the function

```
nltk.ngrams(sequence, n, pad_left=False, pad_right=False,
left_pad_symbol=None, right_pad_symbol=None)
```

```
>>> list(nltk.ngrams(sentence, 2,
    pad_left = True, pad_right = True))
>>> [(None, 'I'), ('I', 'am'), ('am', 'paid'),
('paid', 'by'), ('by', 'the'), ('the', 'word'),
('word', 'so'), ('so', 'I'), ('I', 'always'),
('always', 'write'), ('write', 'the'),
('the', 'shortest'), ('shortest', 'words'),
('words', 'possible'), ('possible', None)]
```

Chain rule

We could resort to the **chain rule** to calculate the probability of a sequence of linguistic entities (characters, syllables, words, sentences).

$$P(A_1, ..., A_k) = P(A_1)P(A_2|A_1)P(A_3|A_1, A_2)...P(A_k| \cap_{i=1}^{k-1} A_i)$$

- P("its water is so transparent") = P("its") * P("water" | "its") * P("is" | "its water") * P("so" | "its water is") * P("transparent" | "its water is so")

How can we estimate these probabilities? Just counting?

- P("transparent" | "its water is so") = f("its water is so transparent") / f("its water is so")

Chain rule, issues

Simply counting the occurrences of long structures (e.g., a sentence) in a corpus is often not a viable option:

1. **data are sparse**; if one term in the chain never occurred in the reference corpus, then the whole sequence is impossible.
2. How can we distinguish the probability of a **rare combination of words** from impossible combinations (a.k.a. agrammatical)? *Colorless green ideas sleep furiously.*
3. Calculating all occurrences of long structures can be **prohibitively expensive** and also not very profitable.

Markov assumption

Instead of calculating the probability of a word given its entire history, we only **approximate** it by using the last few words preceding it. Solves the third problem above, cost effectiveness.

- A Markov assumption of order k: the probability of a word depends only on the last k previous words:

$$P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-k}^{i-1})$$

- For instance:
  - if $k = 0 \Rightarrow$ P("transparent" | "its water is so") P("transparent")
  - if $k = 1 \Rightarrow$ P("transparent" | "its water is so") P("transparent" | "so")
  - if $k = 2 \Rightarrow$ P("transparent" | "its water is so") P("transparent" | "is so")

Markov chain models

A k^{th}-**order Markov model** assumes that each event depends only on a limited window composed by the k preceding events (i.e., we use $k+1$ grams). Higher order models have more "memory", but are more costly to compute.

- Zero-order Markov model: **unigram**.
- First-order Markov model: **bigrams**.
- Second-order Markov model: **trigrams**,

$$P(w) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots P(w_n|w_{n-2}, w_{n-1})$$

Discussion Questions (from Reading Assignment 1)

- Smith highlights a disadvantage of dimensionality reduction is that individual dimensions lose their specific, interpretable meanings. The word's meaning is then distributed across the entire vector rather than being concentrated in specific interpretable features, leading to the term "distributed representations." Are distributed representations therefore a negative consequence of the dimensionality reduction of word vectors?
- Why don't we put only academic literature into a LLM, and train it based on only 'perfectly' spoken language?
- Why is Chomsky opposed to the idea of approaching the thing as a empirical science? From what I understand rationalists wants to make a rigorous structure for language which is understandable as it would mean normalised predictions but languages did not arrive like other sciences, it was made and refined over centuries, so building them rigorously would be quite literally impossible.

More next time...

Estimating n-gram probabilities

Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a method for estimating the parameters of a statistical model (or a probability distribution) by maximizing the likelihood of observed data.

- E.g., if a word has a relative frequency of 0.8 (in a given corpus), we can assume that its probability in language should be 0.8 as well.

Using MLE to estimate bigram probabilities:

$$P(w_i|w_{i-1}) = \frac{f(w_{i-1}, w_i)}{f(w_{i-1})}$$

Maximum Likelihood Estimation

Example with bigrams

- Corpus:
  1. "SoS I am Sam EoS"
  2. "SoS Sam I am EoS"
  3. "SoS I do not like green eggs and ham EoS"
- Some estimated probabilities:
  1. $P(\text{“I”} \mid \text{“SoS”}) = 2/3 = 0.67$
  2. $P(\text{“EoS”} \mid \text{“Sam”}) = 1/2 = 0.5$
  3. $P(\text{“do”} \mid \text{“I”}) = 1/3 = 0.33$

rightarrow Notebook 3, Distributions in text: section Word Frequencies

Data sparsity

N-gram models work well only for those n-grams whose frequency of occurrence in the training corpus is sufficiently high.

- Data sparsity is still an issue: word sequences missing from the data on which we built our model will be treated as zero probability n-grams, notwithstanding their plausibility (remember Chomsky's "grammaticality" objection to probabilistic models).
- The higher the order of the n-gram, the more data sparsity will be an issue.
- **Smoothing** can mitigate this problem by shaving off a bit of probability mass from some more frequent events and give it to rare or unseen events.

rightarrow Notebook 3, Distributions in text: section Zipf's Law

Add-one smoothing

A.k.a. **Laplace smoothing**, solves issue 1 above.

- We pretend we have seen every word once more than we actually have.
- For bigrams:

$$P_{Laplace}(w_i|w_{i-1}) = \frac{f(w_i, w_{i-1}) + 1}{f(w_{i-1}) + |V|}$$

- Usually a bad choice for n-grams, as too much probability mass is moved to all the zeros. It works better for tasks with less sparsity.

Add-one smoothing - I

Credit: J&M, Ch. 3.

	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

¿¿ **Figure 3.5** Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

Add-one smoothing - II

Credit: J&M, Ch. 3.

	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

Figure 3.6 Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

¿¿

Add-k smoothing

- To adjust the quantity of probability mass moved towards the zeros;
- we add a fractional count k instead of 1;
- a dev-test dataset can be used to find the optimal value of k.
- For bigrams:

$$P_k(w_i|w_{i-1}) = \frac{f(w_i, w_{i-1}) + k}{f(w_{i-1}) + k|V|}$$

Backoff and interpolation

Intuition: use less context in place of context you don't have much evidence about.

- **Backoff**: a set of techniques to revert to lower-order sequences when data is not observed. E.g., use trigrams if you have enough data, otherwise use bigrams, or unigrams.
- **Interpolation**: mix trigrams, bigrams and unigrams with weights; it usually works better than backoff.

Linear interpolation

- **Simple linear interpolation**: mixing together the unigram, bigram, and trigram probabilities, each weighted by a positional λ:

  $$\hat{P}(w_i|w_{i-2}, w_{i-1}) = \lambda_1 P(w_i|w_{i-2}, w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i)$$

  Where $\sum_i \lambda_i = 1$
- **Conditioning λ on the context**:

  $$\begin{aligned}\hat{P}(w_i|w_{i-2}, w_{i-1}) = \lambda_1\left(w_{i-2}^{i-1}\right) P(w_i|w_{i-2}, w_{i-1}) + \\ \lambda_2\left(w_{i-2}^{i-1}\right) P(w_i|w_{i-1}) + \lambda_3\left(w_{i-2}^{i-1}\right) P(w_i)\end{aligned}$$

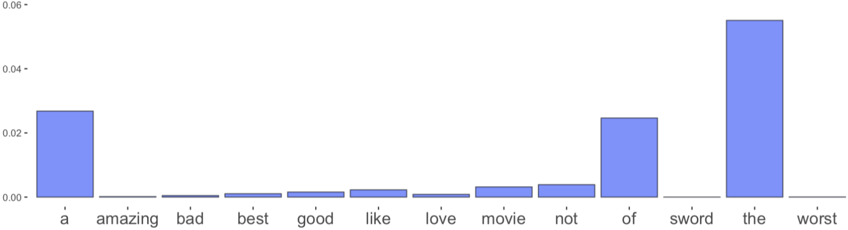
Note: $\hat{P}$ denotes an estimate of P. Yes, notation overloading in action.
Note: how to estimate parameters? Grid search, random search, Expectation Maximization, ..

More smoothing techniques

- "Stupid" backoff: just revert to lower-order n-grams when missing observations. Works pretty well with a lot of data.
- Kesner-Ney smoothing: when backing off to a lower-order n-gram, consider how likely it is for that n-gram to actually show up in a new continuation. Cf. collocations.

Aside 1: Generating - I

Once we have trained our language model, we end up having one **categorical distribution over the vocabulary** for every observed context.



¿¿

Credit: David Bamman (UC Berkeley).

Aside 1: Generating - II

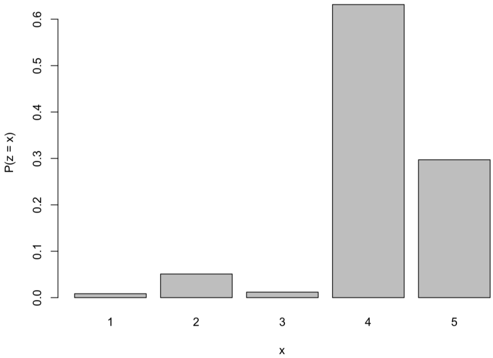
When we generate new samples, we use these distributions **conditioning on the previous context**.

context1	context2	generated word
START	START	The
START	The	dog
The	dog	walked
dog	walked	in

¿¿
Credit: David Bamman (UC Berkeley).

Aside 2: Sampling - I

Consider the probability mass function (PMF), $P(w = x)$

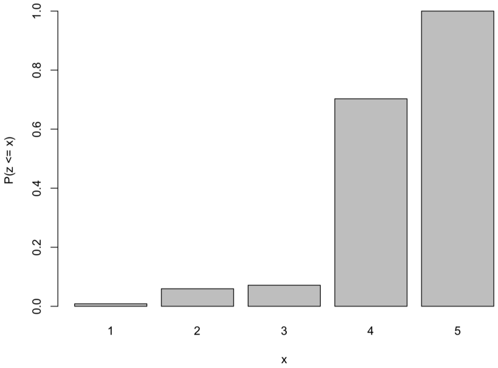


¿¿

Credit: David Bamman (UC Berkeley).

Aside 2: Sampling - II

Consider then the same cumulative density function (CDF), $P(w \leq x)$

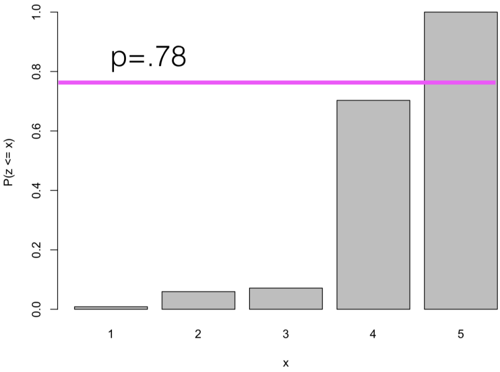


¿¿

Credit: David Bamman (UC Berkeley).

Aside 2: Sampling - III

Sample a ρ uniformly at random over [0,1]. Find the point satisfying $CDF^{-1}(\rho)$.

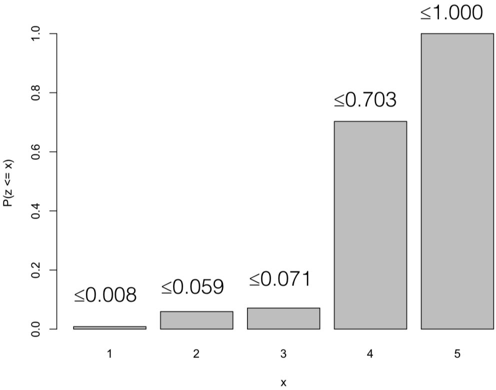


¿¿

Credit: David Bamman (UC Berkeley).

Aside 2: Sampling - IV

When we generate new samples, we use these distributions **conditioning on the previous context**.



¿¿

Credit: David Bamman (UC Berkeley).

Evaluating language models

Approaches to evaluation

- **Extrinsic evaluation**: embed your model in another application (e.g., speech recognizer, machine translation) and see how much it improves. Advantages: practical insight. Limitations: task-specific, time consuming.
- **Intrinsic evaluation**: evaluation of the model independently of its final use. Evaluation strategy: the best model is the one that assigns the highest probability to a previously unseen test set, on the basis of a set of training data. Stated otherwise, *a good language model assigns high probability to real, previously unseen texts*.
    - If we need to set some additional parameter (e.g. for some smoothing techniques), it is useful to test them on a third, development set.
    - Evaluation metric: **perplexity**.

Data splitting

	training	development	testing
size	80%	10%	10%
purpose	training models	model selection; hyperparameter tuning	evaluation; never look at it until the very end

¿¿

Credit: David Bamman (UC Berkeley).

Perplexity

- *Intuition: a good model will give a high probability to real sentences from an unseen test set.* Important constraint: the vocabulary needs to be the same, and the two languages comparable.
- **Perplexity**: the inverse probability of the test set, normalized by the number of words:
  - for a test set composed of n words, $W_{test} = w_1, w_2, \ldots, w_n$:

$$PP(W_{test}) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}} = \sqrt[n]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

  - by applying the chain rule (or using whichever Markov assumption):

$$PP(W_{test}) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i | w_1, \ldots, w_{i-1})}}$$

Perplexity

- We often work with **log probabilities** in NLP, due to the fact that they are usually low: issues of numerical precision rapidly ensue.
- Perplexity can thus be also expressed as follows (unigram and trigram cases):

$$logP(w_1, w_2, \ldots, w_n) = log \prod_{i=1}^{n} P(w_i) = \sum_{i=1}^{n} logP(w_i)$$

$$PP(W_{test}) = exp\Big(- \frac{1}{n} \sum_{i=1}^{n} logP(w_i) \Big)$$

$$PP(W_{test}) = exp\Big(- \frac{1}{n} \sum_{i=1}^{n} logP(w_i | w_{i-2}, w_{i-1}) \Big)$$

Note: the log-sum-exp trick is popular in ML.

Perplexity

- **The lower the better**: minimizing perplexity is equivalent to maximizing the test set probability according to the language model.
- example from J&M. Training set: 38M tokens from the Wall Street Journal ($|V| = 19979$ types); test set: 1.5M tokens.
  - Unigram perplexity: 962.
  - Bigram perplexity: 170.
  - Trigram perplexity: 109.

  These values make most sense in comparison to each other. You can see that there are diminishing returns in considering more context.

There is more to it

- Language models via classification.
- We have not really discussed the issue of closed vocabulary (i.e., avoid probability zero for unseen words).
- Neural language models and large language models (down the line).

No lab on n-gram language modeling, but possible with NLTK.
Recommended notebook → `https://www.kaggle.com/code/alvations/n-gram-language-model-with-nltk`

Linguistic complexity

What kind of sentences and utterances are particularly complex?

Linguistic complexity

- Longer reading times
- Longer eye fixation times
- Slower reaction times
- Slower speech rate
- More errors

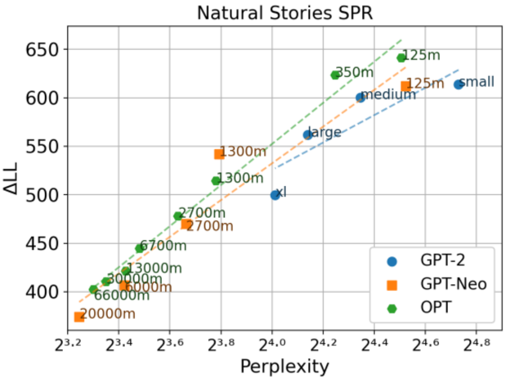
Linguistic complexity

Surprisal theory: More surprising elements are more difficult to process.

- Shannon information theory: Information that requires more bits to encode is more complex
- Events of low probability

Linguistic complexity

Language model perplexity = linguistic complexity?

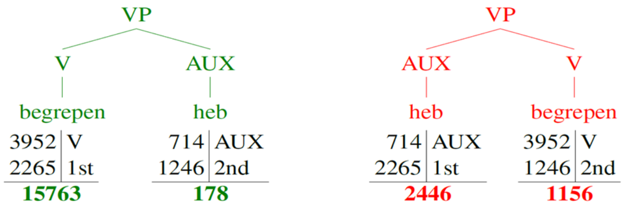


Correlation of GPT and human reading times on Natural Stories corpus (in log-likelihood difference)

Oh & Schuler (2022) - `https://arxiv.org/abs/2212.12131`

Complexity predicts Dutch word order

- *Ik denk dat ik het* **begrepen heb**
- *Ik denk dat ik het* **heb begrepen**



More **uniform information density** in latter word order

Assignments

- Reading assignment Big Picture - done
  - Discussion: next lecture
- Assignment 1: Friday 01/03
- Next reading assignment: 08/03

Elementary probability theory

Probability theory

P.T. is the branch of mathematics dealing with how likely is for an event to happen.

- **Experiment** (a.k.a. a trial): the process by which an observation is made.
  - E.g., tossing three coins;
  - an experiment is a *random experiment* if its outcome is uncertain.
- **Sample Space** (Ω – "omega"): the set of *basic outcomes* (a.k.a. sample points) for our experiment.
  - **Discrete** sample spaces have at most a countably infinite number of basic outcomes (e.g., $\Omega_{tossing} = \{head, tail\}$);
  - **continuous** sample spaces have an uncountable number of basic outcomes (e.g., weight).

Probability theory

- Let an **event** A_i be a subset of the Ω sample space:
  - the *event space* E is the set of all the possible events of the sample space;
  - *elementary (or simple) events* being the unique possible outcomes;
  - an event with more than a sample point is said to be a *complex event*.
- A toy experiment: throwing a dice.
  - **sample space**:
  - **simple event**:
  - **complex event**:
  - **certain event**:
  - **impossible event**:

Probability theory

- Let an **event** A_i be a subset of the Ω sample space:
  - the *event space* E is the set of all the possible events of the sample space;
  - *elementary (or simple) events* being the unique possible outcomes;
  - an event with more than a sample point is said to be a *complex event*.
- A toy experiment: throwing a dice.
  - **sample space**: $\Omega_d = \{1, 2, 3, 4, 5, 6\}$;
  - **simple event**:
  - **complex event**:
  - **certain event**:
  - **impossible event**:

Probability theory

- Let an **event** A_i be a subset of the Ω sample space:
  - the *event space* E is the set of all the possible events of the sample space;
  - *elementary (or simple) events* being the unique possible outcomes;
  - an event with more than a sample point is said to be a *complex event*.
- A toy experiment: throwing a dice.
  - **sample space**: $\Omega_d = \{1, 2, 3, 4, 5, 6\}$;
  - **simple event**: to get a "6", i.e. $A = \{6\}$;
  - **complex event**:
  - **certain event**:
  - **impossible event**:

Probability theory

- Let an **event** A_i be a subset of the Ω sample space:
  - the *event space* E is the set of all the possible events of the sample space;
  - *elementary (or simple) events* being the unique possible outcomes;
  - an event with more than a sample point is said to be a *complex event*.
- A toy experiment: throwing a dice.
  - **sample space**: $\Omega_d = \{1, 2, 3, 4, 5, 6\}$;
  - **simple event**: to get a "6", i.e. $A = \{6\}$;
  - **complex event**: to get an even number, i.e., $A = \{2, 4, 6\}$;
  - **certain event**:
  - **impossible event**:

Probability theory

- Let an **event** A_i be a subset of the Ω sample space:
  - the *event space* E is the set of all the possible events of the sample space;
  - *elementary (or simple) events* being the unique possible outcomes;
  - an event with more than a sample point is said to be a *complex event*.
- A toy experiment: throwing a dice.
  - **sample space**: $\Omega_d = \{1, 2, 3, 4, 5, 6\}$;
  - **simple event**: to get a "6", i.e. $A = \{6\}$;
  - **complex event**: to get an even number, i.e., $A = \{2, 4, 6\}$;
  - **certain event**: to get a number between 1 and 6, i.e., $A = \{1, 2, 3, 4, 5, 6\} = \Omega_d$;
  - **impossible event**:

Probability theory

- Let an **event** A_i be a subset of the Ω sample space:
  - the *event space* E is the set of all the possible events of the sample space;
  - *elementary (or simple) events* being the unique possible outcomes;
  - an event with more than a sample point is said to be a *complex event*.
- A toy experiment: throwing a dice.
  - **sample space**: $\Omega_d = \{1, 2, 3, 4, 5, 6\}$;
  - **simple event**: to get a “6”, i.e. $A = \{6\}$;
  - **complex event**: to get an even number, i.e., $A = \{2, 4, 6\}$;
  - **certain event**: to get a number between 1 and 6, i.e., $A = \{1, 2, 3, 4, 5, 6\} = \Omega_d$;
  - **impossible event**: to get a “7”, i.e., $A = \emptyset$;

Probability

- Probabilities are $[0, 1]$ values, where 0 indicates impossibility and 1 certainty.
- A **probability function** (a.k.a. probability distribution) distributes a probability mass of 1 throughout the sample space Ω,
  - i.e., it assigns a probability P to each event A, denoted as $P(A)$.
- Formally, a probability function is any function $P : E \rightarrow [0, 1]$ such that:
  - $P(\Omega) = 1$;
  - Countable additivity: $P\left(\bigcup_{j=1}^{\infty} A_j\right) = \sum_{j=1}^{\infty} P(A_j)$,

    if $A_j \cap A_i = 0$ for $j \neq k$ (disjoint sets).

Probability

- **Probability (classic definition)**:
  - (“a-priori”) probability is derived by deductive reasoning.
  - Given an experiment whose sample space Ω is finite and where all the sample points are mutually exclusive and equally plausible, for each event A:

$$P(A) = \frac{|A|}{|\Omega|}$$

Probability

Example: trowing a dice

$\Omega_d = \{1, 2, 3, 4, 5, 6\}, |\Omega_d| = 6$.

- to get a "6": $A = \{6\} \hookrightarrow |A| = 1 \hookrightarrow P(A) = |A|/|\Omega_d| = 1/6$;
- to get an even number ..
- to get a number between 1 and 6 ..
- to get a "7" ..

Probability

Example: trowing a dice

$\Omega_d = \{1, 2, 3, 4, 5, 6\}, |\Omega_d| = 6.$

- to get a "6": $A = \{6\} \hookrightarrow |A| = 1 \hookrightarrow P(A) = |A|/|\Omega_d| = 1/6$;
- to get an even number ..
- to get a number between 1 and 6 ..
- to get a "7" ..
- to get 2 tails out of three trows of the dice:
  $\Omega_3 = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}$ ..

Random variables

- A **random variable** is a function $X : \Omega \to \mathbb{R}$. Random variables allow us to work with the probabilities of numerical values that are related to the event space.
  - Random variables can be discrete, when $X : \Omega \to S$, and S is a countable subset of $\mathbb{R}$.
  - Example: events from tossing two dice, a discrete RV could be the sum of their faces. In this case, $S = \{2, ..., 12\}$.
  - The *probability mass function* (pdf) gives the probability of the discrete RV's values. E.g., the probability of the RV value x is $p(X = x)$.
- Note:
  - we mostly work with discrete RVs in NLP;
  - We write $P(A)$ (probability function) and $p(X = x)$ or, simply, $p(x)$ (probability mass function);
  - we can define the **expectation and variance** of RVs ..

Frequentist interpretation

- The a-priori definition of probability doesn't work if Ω is not finite or if events are not equally plausible (e.g., a biased dice).
- According to the **law of large numbers** (LLN), the average of the results obtained by a large number of trials should get close to the expected value, and will get closer as more trials are performed.
- Probability (**frequentist interpretation**):
  - an event's probability is defined as the limit of its relative frequency in a large number of trials;
  - if n is the number of repetitions of an experiment, and f_A is the number of times an event A is observed, then:

$$P(A) = \lim_{n \to \infty} \frac{f_A}{n}$$

Frequentist interpretation

Example: randomly select a word from a corpus

- We can see a corpus as a repeated experiment in choosing n words, over the vocabulary. The probability of selecting each word is likely not uniform, but we don't know it.
- We can estimate the (thus empirical) probability distribution by considering the word frequency. Given f_v the observed frequency of the word v,

$$P(v) = \frac{f_v}{n}$$

Maximum Likelihood Estimation

- **Maximum Likelihood Estimation**: a method for estimating the parameters of a statistical model (or a probability distribution) by maximizing the likelihood of observed data.
  - If a word has a relative frequency of 0.8 (in a given corpus), we can estimate its probability to be 0.8 as well.
- Corpora, however, do not sample the full vocabulary of a language:
  - the relative frequency of **rare words** is not a reliable estimate of their probability;
  - MLE overestimates the probability of the words in the corpus, as the mass is distributed only over the set of words in the corpus ..
  - .. and thus treats non-occurring words as impossible: $P(v \notin V) = 0$, with V the space of events (the vocabulary, in this case).

Parametric vs non-parametric approaches

- Parametric modelling: we assume a parametrized model (e.g., from a family of distributions), and find good parameter values from data.
- Non-parametric modelling: we do not assume any model, and simply estimate probabilities from data.

Smoothing

- General definition: smoothing a dataset means to create an approximating function to allow important patterns to stand out, thus removing noise.
- E.g., we use smoothing to shave off a bit of probability mass from some more frequent events and give it to rare or unseen events.
- **Add-one smoothing**:

$$P_s(v) = \frac{f_v + 1}{n + |V|}$$

  - reminder: $|V|$ is the size of the vocabulary (number of types), n is the length of the corpus (number of tokens);
  - originally due to Laplace (Laplace smoothing);
  - very simple but, generally, not a good choice.

Smoothing

Example: MLE vs smoothed MLE

$n = 80$, $|V| = 30$

- $f(''the'') = 25$
  $f_{MLE}(''the'') = \frac{26}{80} = 0.3125$
  $f_s MLE(''the'') = \frac{26}{110} = 0.2364$
- $f(''bites'') = 1$
  ..

Joint probability

- $P(A, B)$: the probability of events A and B occurring together.
  - the probability of getting two 6s by tossing two dice;
  - the probability of extracting a verb and a noun from a text.
- The joint probability is computed differently depending on the relation between the two events:
  - **Independent events**: the outcome of one event does not influence the probability of the other event;
  - **dependent events**: the outcome of one event influences or affects the probability of the other event. We have to resort to conditional probability.

Joint probability (independent events)

- If two events are independent, the following rule applies: $P(A, B) = P(A)P(B)$.
- Selecting a verb and a noun from a randomly generated text:
  - assuming $n = 80$, $f(noun) = 15$ and $f(verb) = 20$;
  - $P(verb) = 15/80$ and $P(noun) = 20/80$;
  - $P(verb, noun) = P(noun, verb) = 15/80 * 20/80 = 3/16 * 1/4 = 3/64$.

Conditional probability

- $P(A|B)$: the probability of events A happening, given that the event B occurs:
  - a.k.a. the updated probability of the event A given some knowledge;
  - **prior probability**: the probability of an event before considering our additional knowledge;
  - **posterior probability**: the probability of an event that results from using additional knowledge.
- Example, choosing without replacement:
  - in a bag you have 5 green, 5 white and 5 red chips;
  - the first time you draw a chip the probability of getting red one is $1/3$;
  - the second time the probability depends on the result of the first draw;
  - if the first draw was a red chip, then the probability of a red chip is $4/14$;
  - if the first draw was not a red chip, then the probability of a red chip is $5/14$.

Conditional probability

- In general, if the event B has occurred: $P(A|B) = \frac{P(A,B)}{P(B)}$
  - If two events are independent: $P(A|B) = P(A)$
- The multiplication rule: $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$
  - If two events are independent: $P(A, B) = P(A)P(B) = P(B)P(A)$
- The **chain rule** is a generalization of the multiplication rule that allows us to calculate the probability of k joint events:

$$P(A_1, ..., A_k) = P(A_1)P(A_2|A_1)P(A_3|A_1, A_2)...P(A_k| \cap_{i=1}^{k-1} A_i)$$

Bayes' theorem

- The Bayes' theorem lets us swap the order of dependence between events, i.e., to calculate $P(A|B)$ in terms of $P(B|A)$.
  - This is useful when $P(A|B)$ is hard to calculate:

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

  - $P(A|B)$ is called **posterior**: after we have observed B, we know something more about the chance of A.
  - $P(A)$ is called **prior**: the chance of A in absence of any data.
  - $P(B|A)$ is called **likelihood**: the degree of belief in B, having observed A.
  - $P(B)$ is called **evidence**: the probability of observing B.
  - If we think of A as a hypothesis and B as an observation, these labels makes much more sense ..

Bayes' theorem

Example: Using the Bayes' theorem

- Suppose someone comes to you with a new tool to detect a rare form of email spam containing NLP funny jokes, which occurs on average every 200,000 emails.
- If an email belongs to this form of spam, the tool will say so with 0.95 probability (true positive rate).
- If it does not, the tool will say that it does with 0.005 probability (false positive rate).
- It looks pretty good, but just to be sure you do a basic check and suppose that an email turns out positive from the tool's check. How likely it is to be a true NLP funny joke spam email?

Bayes' theorem

Example: Using the Bayes' theorem

- If an email belongs to this form of spam, the tool will say so with 0.95 probability (true positive rate).
- If it does not, the tool will say that it does with 0.005 probability (false positive rate).
- How likely it is to be a true NLP funny joke spam email?

$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

$P(spam) = 1/200.000 = 0.000005$

$P(flagged|spam) = 0.95$

$200.000 * 0.005 = 1000$

$P(flagged) = 1001/200.000 = 0.005005$

$P(spam|flagged) = \frac{(0.95*0.000005)}{0.005005} = 0.00095 = 0.095\%$

Notes

Notes

Notes

Notes