02 - N-grams, Markov Models, Naive Bayes

Stephan Heule, stephan.heule@fhnw.ch

2025-02-21

University of Applied Sciences and Arts
Northwestern Switzerland

Lecture Motivation

Back to natural language processing!

- today we are talking about very “simple” models that are based on counting
- **language models** (LM) are probability models over words or letters
- we also talk about the Markov property, an assumption that makes the problem easier (but oversimplifies)
- Bag of Words (BoW) is introduced (a sparse vector representation)
- we use Naive Bayes to classify text or generate next words

Markov assumption

Next word prediction

Finish this sentence:

I fell…

What is more likely?

- *… up*
- *… down*

- probably the second one → we need probabilities given a context (i.e. previous words)
- using probabilities, this translates into

$$\mathbb{P}[\text{up} \mid \text{I fell}] \leq \mathbb{P}[\text{down} \mid \text{I fell}]$$

Chain rule of a sentence

What is the probability of a whole sentence? [1]

$$\mathbb{P}[\text{I fell down the valley}]$$

$$\mathbb{P}[\text{I fell down the valley}] = \begin{aligned} &\mathbb{P}[\text{I}]\cdot \\ &\mathbb{P}[\text{fell} \mid \text{I}]\cdot \\ &\mathbb{P}[\text{down} \mid \text{I fell}]\cdot \\ &\mathbb{P}[\text{the} \mid \text{I fell down}]\cdot \\ &\mathbb{P}[\text{valley} \mid \text{I fell down the}] \end{aligned}$$

If you had all possible probabilities, you could generate a sentence!

1. use the chain rule: $\mathbb{P}[X_1, \ldots, X_n] = \mathbb{P}[X_1]\mathbb{P}[X_2 \mid X_1]\mathbb{P}[X_3 \mid X_1, X_2] \ldots \mathbb{P}[X_n \mid X_1, \ldots, X_{n-1}]$

The chain rule of probability

Lets abbreviate n words[1] $w_1, \ldots, w_n$ as $w_{1:n}$. Then the chain rule becomes

$$\mathbb{P}[w_{1:n}] = \mathbb{P}[w_1] \prod_{k \geq 2}^{n} \mathbb{P}[w_k \mid w_{1:k-1}]$$

1. I use words and tokens synonymously

Counting

How would you go about to estimate

$$\mathbb{P}[\text{valley} \mid \text{I fell down the}]?$$

Maybe with counting occurrences in a large corpus? The $C(\cdot)$ stands for counting in the training corpus:

$$\mathbb{P}[\text{valley} \mid \text{I fell down the}] = \frac{C(\text{I fell down the valley})}{C(\text{I fell down the})}$$

There are a few challenges:

- a lot of data is needed – especially the larger B gets in $\mathbb{P}[A \mid B]$ → why?
- if nominator is 0 → the whole probability is 0
- what if the denominator is 0?

Shortcut: Markov assumption

Let's relax a little bit at the cost of introducing some variance:

- What if we try an approximation by ignoring "too far away" words/tokens

$$\mathbb{P}[\text{valley} \mid \text{I fell down the}] \approx \mathbb{P}[\text{valley} \mid \text{the}]$$

- What is the benefit of this approximation? What are the tradeoffs?
- A **k-order Markov chain**[1] approximates $w_{1:n}, n > k$ with the k latest tokens: Therefore the test sample *predictable with no fun* is assigned to $C = -$

$$\mathbb{P}[w_n \mid w_{1:n-1}] \approx \mathbb{P}[w_n \mid, w_{n-k}, \dots w_{n-2}, w_{n-1}] = \mathbb{P}[w_n \mid w_{(n-k):(n-1)}]$$

1. For a visualization of general Markov chains

n-grams

A **n-gram** is a sequence of n tokens:

- 2-gram and 3-gram have special terms: **bi-gram**, **tri-gram**

Examples:

- *You are* → bi-gram
- *You are loud* → tri-gram

The **n-gram assumption** uses the last $n-1$ tokens to predict the next one. Thus n-grams relate nicely to Markov chains of order $n-1$

Example with tokens w_i:

- a bigram $\mathbb{P}[w_n \mid w_{1:n-1}] \approx \mathbb{P}[w_n \mid w_{n-1}]$
- a trigram $\mathbb{P}[w_n \mid w_{1:n-1}] \approx \mathbb{P}[w_n \mid w_{n-2}, w_{n-1}]$

The sentence probability for bigrams

Under the bigram assumption, the probability of counting the probabilities get considerably “easier”:

$$\mathbb{P}[w_{1:n}] = \prod_{k=1}^{n} \mathbb{P}[w_k \mid w_{1:k-1}] \approx \prod_{k=1}^{n} \mathbb{P}[w_k \mid w_{k-1}]$$

How can we estimate these probabilities?

- if you have a probability function (e.g. learned from a corpus) you can plug this in
- the easiest way is counting (as before)[1]:

$$\mathbb{P}[w_n \mid w_{n-1}] = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

1. word counts are MLE

Something is missing

Given a corpus with the following sentences

I stand

I fell down

Math is fun, right?

What is the probability of seeing a sentence of length 2 using a bigram approach?

$$\sum_{w_2} \sum_{w_1} \underbrace{\mathbb{P}[w_2 \mid w_1]\mathbb{P}[w_1]}_{=\mathbb{P}[w_1, w_2]} = 1$$

- our probability model assigns all probability mass to sentences of length 2.
- same argument is valid for sentences using 3-grams, etc.
- we want to have one probability distribution, where a sentence can end independently of its length

Start and End of Sentence markers

It turns out, that if we add start and end marks to our corpora, we are safe:

<s> I stand <\s>

<s> I fell down <\s>

<s> Math is fun, right? <\s>

Important

We will do an example in the exercises.

Bi-gram example [1]

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

- data is from “speech understanding project” Berkley Restaurant Project
- count all the occurrences of bigrams
- we see here only a tiny subset, vocabulary space is small $|V| = 1446$
- the full matrix is what we call *sparse* (→ many zeros)

1. taken from Chapter 3 Speech and Language Processing

Bi-gram example continued

Remember: $\mathbb{P}[w_n \mid w_{n-1}] = \frac{C(w_{n-1},w_n)}{C(w_{n-1})}$

We need to normalize words:

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

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

- Could this model generate a sentence like: *I want Chinese food to go*? What if *to go* was never observed?
- → sparseness is an issue

Strategies for removing sparseness

How can we augment the previous frequencies approach such that:

- we can also predict unseen words?
  - e.g. *I want Chinese food to go.*
  - or completely new words that never appeared in the training corpus
- very frequent words do not dominate too much?

Starting point is the frequency table: **manipulate frequencies** and then normalize again.

Strategy I: k-smoothing

- *k-smoothing* is also called *additive smoothing*
- for $k = 1$ it is called *Laplace smoothing* [1]
- idea: add an offset k to the counts:

$$\mathbb{P}[w_n \mid w_{n-1}] = \frac{C(w_n, w_{n-1}) + k}{C(w_{n-1}) + k|V|}$$

- here $|V| = \text{number of disitinct words in the corpus}$
- verify that this is indeed a proper probability (i.e. it sums to one)
- how would you find an appropriate k?

1. The Sunrise Problem

Bi-gram example continued

	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

- now there are no more 0
- within each row, the previously 0's (grey) are now larger than 0, but constant
- is this a good idea?

Backoff

Clearly this holds:

$$C(w_{1:n}) \leq C(w_{1:k}) \text{ for any } k \in 1, \ldots, n-1$$

So why we don't we bootstrap ourselfs by relying recursively on smaller n'-grams in case we encounter a 0 count?

Strategy II: Backoff

For tri-grams:

$$\tilde{\mathbb{S}}[w_1, w_2, w_3] = \begin{cases} \mathbb{P}[w_1, w_2, w_3], & \text{if } \mathbb{P}[w_1, w_2, w_3] > 0 \text{ else} \\ \mathbb{P}[w_2, w_3], & \text{if } \mathbb{P}[w_2, w_3] > 0 \text{ else} \\ \mathbb{P}[w_3], & \text{if } \mathbb{P}[w_3] > 0 \end{cases}$$

This is not a proper probability anymore, thus $\tilde{\mathbb{S}}$ instead of $\mathbb{P}$. If n-gram is not observed, then use (n-1)-gram, etc.

Strategy III: Interpolation

In the “backoff approach” on the previous slide we relaxed to shorter n-grams, when there was a zero evidence – here we combine **all** shorter n-grams. For tri-grams:

$$\begin{aligned}\mathbb{P}[w_n|w_{n-2}w_{n-1}] = \lambda_1\mathbb{P}[w_n]+ \\ \lambda_2\mathbb{P}[w_n|w_{n-1}]+ \\ \lambda_3\mathbb{P}[w_n|w_{n-2}w_{n-1}]\end{aligned}$$

- This to be still a proper probability we have to ensure $\lambda_1 + \lambda_2 + \lambda_3 = 1$[1]
- how would you go about to choose the λ_i?

1. make sure you understand this

Logarithms and exponentials

Numerical evaluation of terms like these:

$$p_1 \times p_2 \times p_3 \times \ldots \text{ for small } p_i > 0$$

can lead to numerical underflow. In practice, you might have to use the exp-log trick:

$$\prod_i p_i = \exp \log \prod_i p_i = \exp \sum_i \log p_i$$

Perplexity, Out of Vocabulary

Perplexity

Is my text generation good?

- gold standard still is human feedback
- suppose you generate tokens from a language model (LM), how to measure if it is good?
- we can use **perplexity** to put a number to it. Given a generated $W = w_{1:n}$

$$
\begin{aligned}
\text{perplexity(W)} &= \mathbb{P}[w_{1:n}]^{-\frac{1}{n}} \\
&= \left(\prod_{i=1}^{n} \mathbb{P}[w_i \mid w_{1:i-1}] \right)^{-\frac{1}{n}} \\
&= \left(\prod_{i=1}^{n} \frac{1}{\mathbb{P}[w_i \mid w_{1:i-1}]} \right)^{\frac{1}{n}}
\end{aligned}
$$

for i=1 we set $\mathbb{P}[w_1 \mid w_{1:-1}] = \mathbb{P}[w_1]$

Perplexity

$$\text{perplexity}(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbb{P}[w_i \mid w_{1:i-1}]}}$$

- let $p_i = \mathbb{P}[w_i \mid w_{1:i-1}]$
- if $p_i \approx 1 \; \forall i \Rightarrow \frac{1}{p_i} \approx 1 \Rightarrow \text{perplexity}(W) \approx 1$
- if some $p_i \approx 0 \Rightarrow \frac{1}{p_i} \gg 1 \Rightarrow \text{perplexity}(W) \gg 1$
- if for some $p_i = 0$ then $\text{perplexity}(W)$ is not defined
- thus you want low perplexity but a perplexity of 1 is actually only possible if all the $p_i = 1$, i.e. a deterministic LM → not very interesting
- measures of how surprised the LM is to observe W

What to do with unseen tokens?

- In language modelling, we often encounter tokens that did not occur in our training set but appear in the test set. These are called **out of vocabulary (OOV)** tokens
- The percentage of OOV words that appear in the test set is called the **OOV-rate**
- In practice you will very often have to deal with OOV

OOV remedies

Adding a pseudo word <UNK>. There are two common ways to train the probabilities of the unknown word model :

1. if you have already a vocabulary V, you can add <UNK> to it and replace all tokens that are not in V with <UNK>
2. if you don't already have a fixed vocabulary V:
   - absolute: choose a threshold value θ and replace all tokens that occur less than θ times with <UNK>
   - relative: choose a vocabulary size[1] Λ and add top Λ tokens to the vocabulary, replacing the rest with <UNK>

1. this is for instance done when training Byte-Pair algorithm.

Naive Bayes and Bag of Words

Recap of last week

Recall Bayes' rule:

$$\mathbb{P}[A \mid B] = \frac{\mathbb{P}[B \mid A]\,\mathbb{P}[A]}{\mathbb{P}[B]}$$

Coming back to a classification task with feature $\mathbf{x}_i$ and true class $y_i = c_i$. We see

$$\begin{aligned}\hat{c}_i &= \mathbf{argmax}_c \mathbb{P}[c|\mathbf{x}_i] \\ &= \mathbf{argmax}_c \frac{\mathbb{P}[\mathbf{x}_i \mid c]\,\mathbb{P}[c]}{\mathbb{P}[\mathbf{x}_i]} \\ &= \mathbf{argmax}_c \mathbb{P}[\mathbf{x}_i \mid c]\,\mathbb{P}[c]\end{aligned}$$

Why is the last expression true?

Sentiment analysis example

- The task of classifying a text or sentence into positive or negative classes is called **Sentiment Analysis**.
- This is a classification problem with two classes $C_1 = +$ and $C_2 = -$:
  - *This was a great movie* belongs to $C1$
  - *Worst experienced ever!* to C_2

Question: I give you a large corpus $\{\mathbf{x}_i, c_i\}_{i=1}^N$ with $N \gg 0$ and $c_i \in \{1, \ldots, k\}$[1]. How would you estimate $\mathbb{P}[c_i]$?

1. that is we have k classes

Sentiment analysis example

Our goal is to estimate the probability of $\mathbf{x}$ having positive or negative sentiment, i.e.

$$\hat{c}_i = \mathbf{argmax}_c \mathbb{P}[\mathbf{x} \mid c] \, \mathbb{P}[c]$$

- estimating the class **prior probability** $\mathbb{P}[c_i]$ can be done by counting, i.e. $\mathbb{P}[c_i] = \frac{C(c_i)}{k}$ where $k = \text{number of classes}$
- it remains to estimate the **likelihood** part $\mathbb{P}[\mathbf{x} \mid c]$

Naive Bayes assumption

Naive Bayes is called *naive* because it naively assumes that the features are *conditional independent*[1] of each other:

$$\begin{aligned}\mathbb{P}[\mathbf{x} \mid c] &= \mathbb{P}[x_1 \mid c]\,\mathbb{P}[x_2 \mid c] \ldots \mathbb{P}[x_n \mid c] \\ &= \prod_{i=1}^{n} \mathbb{P}[x_i \mid c]\end{aligned}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$

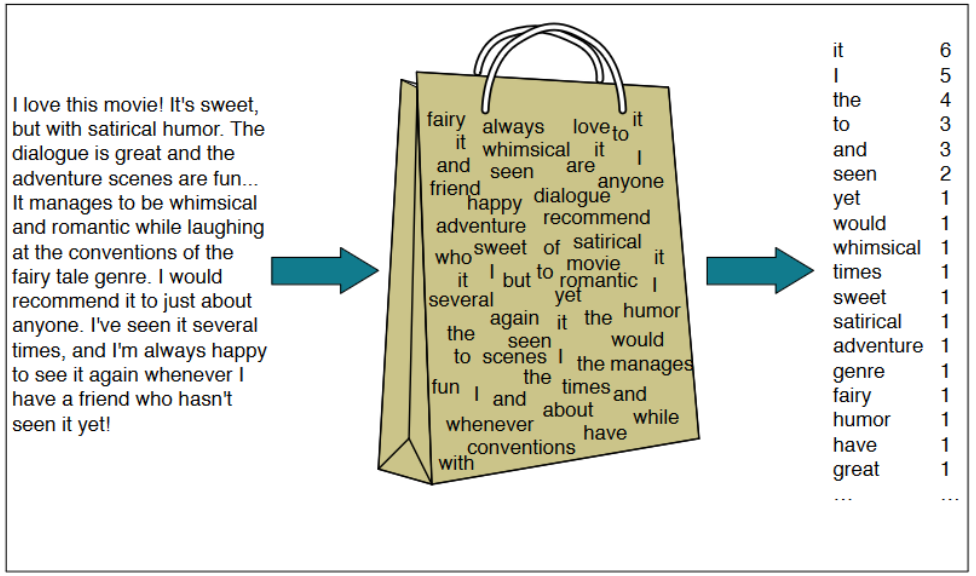
Naive Bayes is often good baseline model – how you model the $\mathbb{P}[x_i \mid c]$ is up to you (e.g. through very basic counting or more complex modelling strategies)

1. Conditional independence

Naively ignoring word order

- Conditional independence for words means that we ignore the ordering
- the product $\prod_{i=1}^{n} \mathbb{P}[x_i \mid c]$ does not depend on the order you multiply the numbers, i.e. $p_1 \times p_2 = p_2 \times p_1$
- that is what **Bag of Words** does

Bag of words



Bag of Words and is easiest described by an image (source Chapter 4 in Speech and Language Processing)

- on the left side we start with a text and on the right side we end up with a vector
- more precisely the vector associated with the text is $(6, 5, 4, 3, 3, \ldots)$. The numbers are the counts of the associated words
- in the bag of words approach, we loose the ordering
- this is a **sparse vector representation**

Example: Bag of Words

1. Write down a bag of word representation of *Man bites dog* and also for *Dog bites man*. Any thoughts?
2. Given the vocabulary $V = \{w_1, \ldots, w_N\}$. How large will the representation of the word *Now* be given that *Now* is in V?

Estimating the likelihood $\mathbb{P}[\mathbf{x} \mid c]$

- as in the n-grams example, we count!
- let $\mathbf{x} = (x_1, \ldots, x_n)$ be some text (i.e. x_i are tokens) that belongs to class C_k
- then counting leads to

$$\hat{\mathbb{P}}[x_i \mid C_k] = \frac{C(x_i \mid C_k)}{\sum_{x \in V} C(x \mid C_k)}$$

- where $C(x_i \mid C_k)$ are the counts of word x_i in class C_k
- do you recall the problems with “plain” counts from earlier?

Problem

- if one expression in the likelihood is 0 so is the complete expression. This is problematic.
- two cases:
  1. new words in *test-data*
  2. there is a word w that does **not** belong to some class C_i
- remedies:
  1. delete new tokens that have not been in the training data
  2. apply **k-smoothing**

logarithms

Recall that multiplying small numbers might result in underflow. In order to evaluate this expression $\operatorname{argmax}_c \mathbb{P}[\mathbf{x} \mid c]\, \mathbb{P}[c]$

we can equivalently do this:

$$c_{\text{NB}} = \arg\max_{c \in C} \left(\log \mathbb{P}[c] + \sum_{w_i \in V} \log \mathbb{P}[w_i | c] \right)$$

Example: Naive Bayes

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Example 4.3 in Speech and Language Processing

1. what is the complete vocabulary in the *Training* data?
2. what are the class priors?
3. are there words in *Test* that don't occur in *Training*
4. calculate the word probabilities in the training data
5. calculate $c_{\text{NB}} = \arg\max_{c \in C} \left(\log \mathbb{P}[c] + \sum_{w_i \in V} \log \mathbb{P}[w_i|c] \right)$ for the test data. Is the test sentence positive or negative?

Example: Naive Bayes

1. Vocabulary

$$V = \{\text{just}, \text{plaing}, \text{boring}, \text{entirely}, \text{predictable}, \\ \text{and}, \text{lacks}, \text{energy}, \text{no}, \text{surprises}, \text{few}, \text{laughs}, \\ \text{powerful}, \text{the}, \text{most}, \text{fun}, \text{film}, \text{of}, \text{summer}\}$$

2. Class priors:

$$\mathbb{P}[+] = 2/5, \quad \mathbb{P}[-] = 3/5$$

3. Yes, we remove *with*. It is not part of the training data

Example: Naive Bayes

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \quad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$
$$P(\text{“no”}|-) = \frac{1+1}{14+20} \quad P(\text{“no”}|+) = \frac{0+1}{9+20}$$
$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \quad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. The word probabilities from training data.

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$
$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

5. The sentiment is negative (lager probability for negative class).

Remarks

- **binary naive Bayes**:
  - often it is more important that a word is present than how often
  - thus restricting counts to 1 improves often prediction
- *negation*:
  - words like *not, never, n't, no* change the meaning of a sentence
  - e.g. *I didn't like this movie* is almost the same as *I did like the movie*
  - what you can do as a baseline: generate new tokens by prepending *NOT*, e.g. *I didn't NOT_like NOT_this NOT_movie*
  - *NOT_like, …* are new tokens
- use *sentiment lexicons* with annotated words for $+/-$ classes:
  - these are word lists that have a positive/negative connotation, e.g. *good, like, …* or *bad, ugly, …*
- Naive Bayes can also be used for language modelling

Summary

- What are language models, n-grams? What is the Markov assumption?
- Use the chain rule to express a sentence probability and describe the shortcut using n-grams and Markov.
- Why do we need a start and end of sentence token?
- What problem address k-smoothing, Backoff, Interpolation? What OOV? And how?
- What is perplexity and what properties does it have?
- Naive Bayes and binary naive Bayes: Why are they naive? What kind of problems can you use them on? What is the difference between the two?
- What is class prior and what is the likelihood?
- How is Bag of Words transforming words into numbers? Is it sparse or dense and why?