Last session: an algorithmic solution to sentiment detection

- You built a symbolic system.
- The information source in your system was the sentiment lexicon.
- It was based on human intuition and required much human labour to build.
- You evaluated it in terms of accuracy.
- Accuracy is an adequate metric because the data was balanced.
- Is there a way to achieve a higher accuracy?
We will start today with a simple machine learning (ML) application.

Definition of ML: a program that learns from data, i.e., adapts its behaviour after having been exposed to new data.

Hypothesis: we can learn which words (out of all words we encounter in reviews) express sentiment rather than relying on a fixed set of words decided independently from the data and before the experiment (sentiment lexicon approach).
Two tasks in ML – classification vs prediction

- **Classification**: Which class (label) should the data I see have?
  - This is what we are doing here.
- **Prediction**: Which data is likely to occur in the given situation?
Features and classes

- Input: easily observable data [often not obviously meaningful] – features $f_i$ (or observations $o_i$)
- Output: meaningful label associated with the data [cannot be algorithmically determined] – class $c_n$
- Classification algorithm is a function that maps from features $f_i$ to target class $c_n$
Your system from Task 1 is already a classification algorithm, but it’s not an ML algorithm.

A statistical classifier maximises the probability that a class $c$ is associated with the observations $o$, and returns the maximising class $\hat{c}$:

$$\hat{c} = \arg\max_{c \in C} P(c|o)$$

$c$ is a class $c \in C = \{c_1 \ldots c_m\}$, the set of classes.

In our case, the observations $o$ are the entire document $d$. 
A machine learning algorithm has two phases: training and testing.

**Training**: the process of making observations about some known data set
- You are allowed to manipulate the $f_i$ (and maybe look at $c_n$ while you do that)

**Testing**: the process of applying the knowledge obtained in the training stage to some new, unseen data

Important principle: never test on data that you trained a system on
Supervised vs unsupervised ML

- **Supervised ML**: you use the classes that come with the data in the training and the testing phase.

- **Unsupervised ML**: you use the classes only in the testing phase.
Naive Bayes Classifier

\[ c_{NB} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} P(c) \prod_{i \in \text{positions}} P(w_i|c) \]

Document \( d \) is represented by word positions \( w_i \), the word encountered at position \( i \) in the test document; \( \text{positions} \) is the set of indexes into the words in the document.

- In the **training** phase, you will collect whatever information you need to calculate \( P(w_i|c) \) and \( P(c) \).
- In the **testing** phase, you will apply the above formula to derive \( c_{NB} \), the classifier’s decision.
- This is supervised ML because you use information about the classes during training.
How did we get from
\( \hat{c} = \arg\max_{c \in C} P(c|d) \)
to
\( c_{NB} = \arg\max_{c \in C} P(c) \prod_{i \in \text{positions}} P(w_i|c) \)?

We got there in three steps:

- **Bayes’ Rule:**
  \[ P(c|d) = \frac{P(c)P(d|c)}{P(d)} \]

- **\( P(d) \)** does not affect \( \hat{c} \)

- **Independence assumption:**
  \[ P(w_1, w_2, ..., w_n|c) = P(w_1|c) \cdots P(w_2|c) \times \cdots \times P(w_n|c) \]
Data Split

- From last time, you have 1800 documents which you used for evaluation.
- We now perform a data split into 200 for testing, 1600 for training.
- You may later want to compare how well the NB System is doing in comparison to the symbolic system.
  - As the NB system is evaluated only on 200 documents.
  - Therefore, you should rerun your symbolic system on the same 200 documents.
Maximum Likelihood Estimates (MLE) $\hat{P}(w_i|c), \hat{P}(c)$

- Maximum Likelihood estimation (MLE) = finding the parameter values that maximize the likelihood of making the observations given the parameters

\[
\hat{P}(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}
\]

\[
\hat{P}(c) = \frac{N_c}{N_{doc}}
\]

- $N_c$: number of documents with class $c$
- $N_{doc}$: total number of documents
- $\text{count}(w_i, c)$: number of word positions $w_i$ occurring together with a class $c$
- $V$: vocabulary of distinct words
A problem you might run into

- A certain word may not have occurred together with one of the classes in the training data, so the count is 0.
- Part of your task today:
  - understand why this is a problem
  - work out what you could do to deal with it
Your task for today

Task 2:

- Write code that calculates the MLE $\hat{P}(w_i|c)$ and $\hat{P}c$, using only the training set.
- Now you have covered the training phase.
- Then write code for testing, i.e., apply your classifier to the validation set.
- Measure accuracy on the 200 documents.
- When you design your data structures, you may want to consider that you will in later sessions dynamically split data into a training and test set.
Ticking today

- Task 1 – Symbolic Classifier
Literature

- Textbook Jurafsky and Martin Edition 2, Chapter 6.2: Naive Bayes Classifier