Command line completion (CLC)

an illustration of learning and decision making
using the imprecise Dirichlet model

Erik Quaeghebeur
Classical CLC in action

login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ _
Classical CLC in action

login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1

```
command-prompt$ log<TAB>
logger login logname logout
command-prompt$ log
```
Classical CLC in action

login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger   login   logname   logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ _
Classical CLC in action

login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1

command-prompt$ log<TAB>
logger  login  logname  logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ ls<ENTER>
mail/  logic.dvi  logic.tex
command-prompt$ _
Classical CLC in action

login: erik
Password: 
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger login logname logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ ls<ENTER>
mail/ logic.dvi logic.tex
command-prompt$ dvips log_
Classical CLC in action

login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger login logname logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ ls<ENTER>
mail/ logic.dvi logic.tex
command-prompt$ dvips log<TAB>
logic.dvi  logic.tex
command-prompt$ dvips logic.d<TAB>
command-prompt$ dvips logic.dvi _
Properties of classical CLC

- Two completion action types:
  - list the possible completions, or
  - return the unique completion.
Properties of classical CLC

Two completion action types:
- list the possible completions, or
- return the unique completion.

Rule-based:
- allows for context dependency, and
- requires a categorized database of commands.
Properties of classical CLC

- Two completion action types:
  - list the possible completions, or
  - return the unique completion.

- Rule-based:
  - allows for context dependency, and
  - requires a categorized database of commands.

- User independent:
  - reliable, but
  - does not take command history into account.
Complementing classical CLC

We want to take the command-history into account:

- Whenever there are multiple completions possible.
Complementing classical CLC

We want to take the command-history into account:

- Whenever there are multiple completions possible.
- By building and updating a model for the user’s behavior.
Complementing classical CLC

We want to take the command-history into account:

- Whenever there are multiple completions possible.
- By building and updating a model for the user’s behavior.
- To add completion action types, such as
  - returning the ‘best guess’ completion on the command line,
  - listing a set of ‘best guesses’,
  - listing all possible completions, but ordered.
The set of possible completions

Two illustrative completions:

- command-prompt$ ha<TAB>
  halt  hash

- command-prompt$ pin<TAB>
  pine  ping  pinky
The set of possible completions

Two illustrative completions:

command-prompt$ ha<TAB>
  halt    hash

⇒ Ωha = {halt, hash} ⊋ ωha

command-prompt$ pin<TAB>
  pine    ping    pinky

⇒ Ωpin = {pine, ping, pinky} ⊋ ωpin
The user as a multinomial process

Model of the user’s behavior:

- A priori, there is a fixed probability $t_{\text{command}}$ for every command.
The user as a multinomial process

Model of the user’s behavior:

- A priori, there is a fixed probability $t_{\text{command}}$ for every command.

- After typing part of a command, the remaining possible completions are chosen with the corresponding conditional probabilities.
The user as a multinomial process

Model of the user’s behavior:

- A priori, there is a fixed probability $t_{\text{command}}$ for every command.

- After typing part of a command, the remaining possible completions are chosen with the corresponding conditional probabilities.

Graphical representation of a user:

$$(t_{\text{halt}}, t_{\text{hash}}) = \left(\frac{1}{4}, \frac{3}{4}\right)$$
The user as a multinomial process

Model of the user’s behavior:

- A priori, there is a fixed probability $t_{\text{command}}$ for every command.
- After typing part of a command, the remaining possible completions are chosen with the corresponding conditional probabilities.

Graphical representation of a user:
The user as a Markov process

Model of the user’s behavior:

- A priori, there is a fixed probability $t_{\text{command|previous}}$ for every command and every previously typed command.
The user as a Markov process

Model of the user’s behavior:

- A priori, there is a fixed probability \( t_{\text{command}|\text{previous}} \) for every command and every previously typed command.

- After typing part of a command, the remaining possible completions are chosen with the corresponding conditional probabilities for the previous command.
The user as a Markov process

Model of the user’s behavior:

- A priori, there is a fixed probability $t_{\text{command}|\text{previous}}$ for every command and every previously typed command.
- After typing part of a command, the remaining possible completions are chosen with the corresponding conditional probabilities for the previous command.

Graphical representation of a user:
Knowledge about the user’s behavior

Three models:

- An exact model: $t_{\text{command}}$ is known for all commands.
Knowledge about the user’s behavior

Three models:

- An exact model: $t_{\text{command}}$ is known for all commands.
- A precise Dirichlet model (PDM): the uncertainty about the exact model is determined by a Dirichlet distribution.

\[
\text{Di}(\vec{\vartheta} \mid h, \vec{t})
\]

hal t

\[\rightarrow\]

hash
Knowledge about the user’s behavior

Three models:

- An exact model: $t_{\text{command}}$ is known for all commands.
- A precise Dirichlet model (PDM): the uncertainty about the exact model is determined by a Dirichlet distribution.

\[
\tilde{t} = P_{\text{Di}}(\tilde{\vartheta} \mid h, \tilde{t})
\]

\[
P_{\text{Di}}(X \mid h, \tilde{t}) = \int_{\Delta_{\text{ha}}} X(\tilde{\vartheta})Di(\tilde{\vartheta} \mid h, \tilde{t})d\tilde{\vartheta}
\]
Knowledge about the user’s behavior

Three models:

- An exact model: $t_{\text{command}}$ is known for all commands.
- An *imprecise Dirichlet model* (IDM): the uncertainty is determined by a set of Dirichlet distributions.

![Diagram](image)
Knowledge about the user’s behavior

Three models:

- An exact model: $t_{\text{command}}$ is known for all commands.
- An \textit{imprecise Dirichlet model} (IDM): the uncertainty is determined by a set of Dirichlet distributions.

\[ \bar{t} = P(\vec{\vartheta} \mid h, T) \quad \underline{t} = \overline{P}(\vec{\vartheta} \mid h, T) \]
\[ P_{\text{Di}}(X \mid h, T) = \inf_{\vec{t} \in T} P_{\text{Di}}(X \mid h, \vec{t}) \]
\[ \overline{P}_{\text{Di}}(X \mid h, T) = \sup_{\vec{t} \in T} P_{\text{Di}}(X \mid h, \vec{t}) \]
Observations, Sufficient statistics, and . . .

Observations:
- (a sequence of) executed commands for the multinomial model, or
- (a sequence of) consecutively executed commands for the Markov model.
Observations, Sufficient statistics, and . . .

- Observations:
  - (a sequence of) executed commands for the multinomial model, or
  - (a sequence of) consecutively executed commands for the Markov model.

- Keep what’s relevant for the model: sufficient statistic, the number of occurrences of the commands $\vec{n}$, or the number of occurrences of a transition between commands $N$. 
...Likelihood functions

- **Likelihood function**: likelihood of an exact model given the observations,
  - a multinomial distribution $L_{\vec{n}}(\vec{\vartheta})$, or
  - a Whittle distribution $L_{N}(\Theta)$, proportional to the product of the $L_{\vec{n}}(\vec{\vartheta})$ for each of the previous commands.
Learning using a PDM/IDM

- Updating a Dirichlet distribution using Bayes’ rule:

\[
f(\vec{\vartheta} \mid h, \vec{t}, \vec{n}) = \frac{\text{Di}(\vartheta \mid \vec{h}, \vec{t})L_{\vec{n}}(\vec{\vartheta})}{P(L_{\vec{n}} \mid h, \vec{t})} = \text{Di}(\vartheta \mid h_n = h + n, \vec{t}_n = \frac{ht + \vec{n}}{h + n}).
\]
Learning using a PDM/IDM

Updating a Dirichlet distribution using Bayes’ rule:

\[
f(\vec{\vartheta} | h, \vec{t}, \vec{n}) = \frac{\text{Di}(\vartheta | \vec{h}, \vec{t}) L_{\vec{n}}(\vec{\vartheta})}{P(L_{\vec{n}} | h, \vec{t})}
\]

\[
= \text{Di}(\vartheta | h_n = h + n, \vec{t}_n = \frac{ht + n}{h + n}).
\]

Graphically:
Learning using a PDM/IDM

Updating a PDM is updating the underlying distribution:

\[ P(X \mid h, t) \xrightarrow{n} P(X \mid h_n, t_n). \]
Learning using a PDM/IDM

- Updating a PDM is updating the underlying distribution:

\[ P(X | h, \vec{t}) \xrightarrow{\vec{n}} P(X | h_n, \vec{t}_n). \]

Graphically:
Learning using a PDM/IDM

Updating an IDM comes down to updating the corresponding (set of) PDM's:

\[
P(X \mid h_n, T_n) = \inf \{ P(X \mid h_n, \tilde{t}_n) \mid h_n = h + n, \tilde{t}_n = \frac{ht + \bar{n}}{h + n}, \tilde{t} \in T \}.
\]
Learning using a PDM/IDM

- Updating an IDM comes down to updating the corresponding (set of) PDM’s:

\[
P(X \mid h_n, T_n) = \inf\{P(X \mid h_n, t_n) \mid h_n = h + n, t_n = \frac{ht + n}{h + n}, t \in T\}.
\]

Graphically:
Learning using a PDM/IDM

Updating an IDM comes down to updating the corresponding (set of) PDM’s:

\[
P(X \mid h_n, T_n) = \inf \{ P(X \mid h_n, \vec{t}_n) \mid h_n = h + n, \vec{t}_n = \frac{h\vec{t} + \vec{n}}{h + n}, \vec{t} \in T \}.
\]

Graphically:
Actions and (expected) utility

Each completion action has a certain utility for the user (part of the model of the user). Let a gamble $X_a$ correspond to each action $a$. 
Actions and (expected) utility

- Each completion action has a certain utility for the user (part of the model of the user). Let a gamble $X_a$ correspond to each action $a$.

- The expected utility of an action:
  - when a PDM is used: $P(X_a \mid h, t)$,
  - when an IDM is used: $\underline{P}(X_a \mid h, T)$ and $\overline{P}(X_a \mid h, T)$. 
Actions and (expected) utility

Each completion action has a certain utility for the user (part of the model of the user). Let a gamble $X_a$ correspond to each action $a$.

The expected utility of an action:
- when a PDM is used: $P(X_a | h, \vec{t})$,
- when an IDM is used: $P(X_a | h, T)$ and $\overline{P}(X_a | h, T)$.

Choosing one action instead of another also has an expected utility. This is the prevision of the difference of the corresponding gambles ($P(X_a - X_b)$ or $\overline{P}(X_a - X_b)$).
Decision making: choosing an action

Ordering the actions based on the expected utility of choosing one action over the other.
Ordering the actions based on the expected utility of choosing one action over the other.

When using a PDM:

- compare the actions:
  
  \[ a \succ b \iff P(X_a - X_b) > 0 \iff P(X_a) > P(X_b), \]

- create an ordering of the actions,

- identify the maximal action(s)

  \[ a \text{ is maximal} \iff \forall b : P(X_a) \geq P(X_b). \]
Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.

- PDM: complete ordering of actions.
Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.

- PDM: complete ordering of actions.

- When using an IDM:
  - compare the actions:
    \[ a \succ b \iff \mathbb{P}(X_a - X_b) > 0 \iff \mathbb{P}(X_b - X_a) < 0, \]
  - create an ordering of the actions,
  - identify the maximal action(s):
   \[ a \text{ is maximal} \iff \forall b : \mathbb{P}(X_a - X_b) \geq 0. \]
Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.

- PDM: complete ordering of actions.

- IDM: partial ordering of actions.
Choosing an action

- Choosing the maximal action:
  - Is there a unique maximal action?
  - If there is a unique maximal action: choose it (e.g., returning a completion).
Choosing an action

- Choosing the maximal action:
  - Is there a unique maximal action?
  - If there is a unique maximal action: choose it (e.g., returning a completion).

- The need for a default action:
  - Whenever there isn’t a maximal action, then
  - choose this action (e.g., list the actions according to the ordering).
IDM-CLC in action

login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ _
IDM-CLC in action

login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2

```
command-prompt$ pin<TAB>
pine  pinky
    ping
command-prompt$ pin_
```
IDM-CLC in action

login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine pinky
    ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ _
IDM-CLC in action

login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine  pinky
    ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ pin<TAB>
command-prompt$ pinky _
IDM-CLC in action

login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine  pinky
    ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ pin<TAB>
command-prompt$ hash<ENTER>
...[some text] ...
command-prompt$ _
IDM-CLC in action

login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2

```
command-prompt$ pin<TAB>
pine   pinky
        ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ pin<TAB>
command-prompt$ hash<ENTER>
...[some text] ...
command-prompt$ pin<TAB>
ping    pine
        pinky
command-prompt$ pin_
```