COMMAND LINE COMPLETION:
AN ILLUSTRATION OF LEARNING AND DECISION MAKING
USING THE IMPRECISE DIRICHLET MODEL

ERIK QUAEGHEBEUR & GERT DE COOMAN

SYSTeMS Research Group
Department of Electrical Energy, Systems & Automation, Ghent University

Technologiepark 914, B-9052 Zwijnaarde, Belgium

{Erik.Quaeghebeur,Gert.deCooman}@UGent.be

UNIVERSITEIT GENT

FACULTEIT TOEGEPASTE WETENSCHAPPEN

I. INTRODUCTION

Purpose Illustrate how models using imprecise probabilities (a generalization of standard probabilities [1]) can be used in a practical application.

Application We've chosen command line completion because it's

- conceptually simple,
- relevant for daily life (making computers more user friendly),
- ideal for showing the important aspects of the models we're investigating.

Models Two probabilistic models are used in this illustration:

- The imprecise Dirichlet model or IDM, used for representing probabilistic information about multinomial processes [2]. We combine it with a complementary updating scheme, so we can learn from observations. We can use it for decision making when utility specifications for completion actions are given. It has already been used for other applications [3], [4].
- Our research is concerned with using the IDM for learning in Markov models. This will be incorporated in this illustration to allow the completion method to take preceding commands into account.

II. CONCEPTS AND MODELS

We first look at the main concepts and at the probabilistic models that we use.

A. Command line completion

Where it is used It's a feature of operating systems (OS) provided for users working in text mode.

How to use it The user types part of a command (doesn't know the rest or too lazy to type the rest) and can then press a predefined key (here: <TAB>). Then, the OS shows a list of n possible completions i or returns the unique completion. An example with four ($n = 4$) possible completions is given below:

```
command-prompt# log<TAB>
logger    login    logname    logout
command-prompt# logn<TAB>
command-prompt# logname_
```

Our objective Add supplementary functionality when multiple completions are possible. We'll model the behavior of the user, so the OS can order the possible completion actions, which can be

- to return a 'best guess' completion,
- to present the possible completions in a for the user usefully ordered fashion.

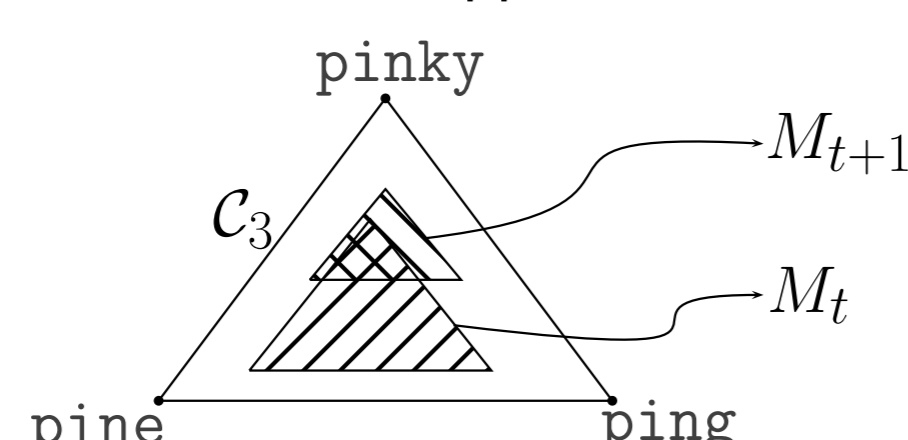Similar work has been done as a testbed for web prefetching techniques [5].

B. The imprecise Dirichlet model

Basic assumption When choosing a completion the user's behavior can be modelled using a multinomial distribution c, which is fixed but unknown. A component c_i gives the chance that the user chooses completion i. The completion simplex C_n is the set of all possible multinomial distributions. This simplex is shown for the completions of 'pin':

$$\mathcal{C}_3 \qquad c = (\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3})$$

pinky, pine, ping
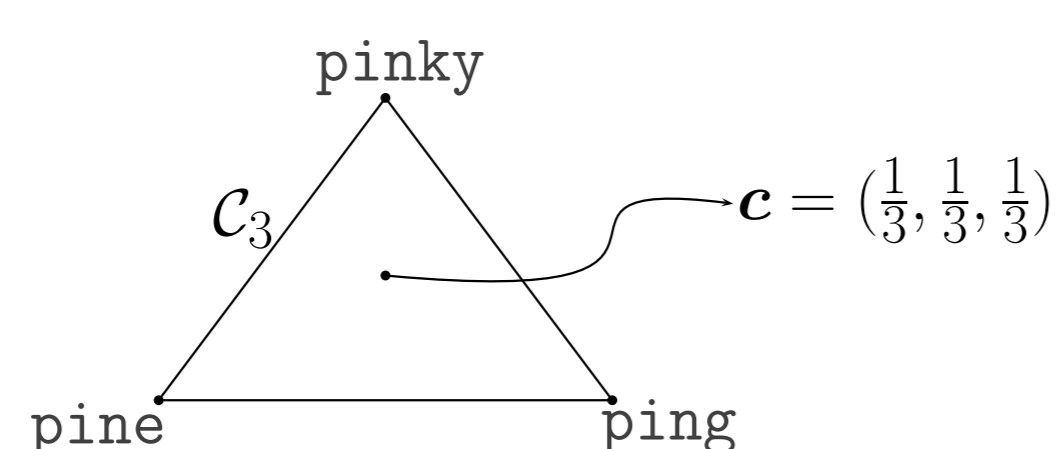
Representing what we know The OS uses an imprecise Dirichlet model to represent the partial, probabilistic knowledge about c it can obtain. This model consists of a convex set of Dirichlet distributions over the completion simplex C_n. The expectation of c under this model consists of a subset M_t of C_n.

Updating our knowledge The OS updates the IDM after each observation (each time the user enters a command). After updating, the new expectation M_{t+1} will be more precise. Below, you can see what happens after observing 'pinky':

$$\mathcal{C}_3 \qquad M_{t+1} \qquad M_t$$

pinky, pine, ping

Initially, we don't know a thing, so M_0 is taken to be the whole interior of C_n.

C. Markov models: extending the IDM

Additional assumption When we take into account that the user's choice of a completion can depend on the command(s) typed on the preceding command line(s), the multinomial distribution c will depend on the preceding command(s).

Additional model We use a Markov model to formalize this behavior. It consists of a transition matrix T, which is fixed but unknown. The jth row of T will correspond to the multinomial distribution, conditional on the preceding command(s). For the completions of 'pin', (part of) a possible T is shown below:

$$i = \text{pine, ping, pinky} \qquad j =$$
$$T = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/4 & 1/4 \end{pmatrix} \begin{matrix} \text{logname} \\ \text{logger} \end{matrix}$$

The new representation The OS then uses a modified version of the IDM, using one set of Dirichlet distributions per row of T. Updating can be done after observing a transition, which corresponds to two (or more) successive commands.

III. ACTIONS, UTILITY AND DECISION MAKING

Now that we have a model for the behavior of the user and we can keep it up-to-date, it can be used to decide what completion actions to take.

A. Completion actions and utility

Possible actions A set of actions can consist of, but is not limited to

- presenting a completion i on the command line: action a_i.
- giving a specially ordered list of the n possible completions: the default action.

These two types of actions are illustrated below:

```
command-prompt# pin<TAB>        command-prompt# log<TAB>
command-prompt# pine_           logout    logger
                                          login
                                          logname
```

Utility of an action The user is looking for a specific completion, so each non-default action taken by the OS has a certain utility for the user. For example, action a_i has a utility of $+1$ if i is the correct completion, and -1 if it isn't.

B. Expected utility and decision making

Estimating the user's benefit By combining the utility of an action with the model of the user's behavior (the IDM), the OS can calculate an expected utility for each of the actions.

Choosing an action The expected utilities can be used to construct a partial ordering of the actions a_i. If there is a maximal element in the ordering, that action is taken. If not, a safe default is chosen: showing the partial order to the user.

IV. CONCLUSIONS

We've given a description of a method for command line completion based on probabilistic models that allow for learning from observation and cautious decision making. The described models can also be used in conceptually similar situations, such as

- the prefetching of web pages,
- word completion for the SMS function of mobile phones.

An at first sight unrelated field as bio-informatics also contains possible applications, such as the alignment of gene sequences, which will be part of future research.

REFERENCES

[1] Peter Walley, *Statistical reasoning with imprecise probabilities*, Chapman and Hall, London, 1991.

[2] Peter Walley, "Inferences from multinomial data: learning about a bag of marbles," *Journal of the Royal Statistical Society* B, vol. 58, no. 1, pp. 3–57, 1996.

[3] Marco Zaffalon, "The naive credal classifier," *Journal of Statistical Planning and Inference*, vol. 105, no. 1, pp. 5–21, 2002.

[4] Erik Quaeghebeur and Gert de Cooman, "Game-theoretic learning using the imprecise Dirichlet model," in *ISIPTA'03 – Proceedings of the Third International Symposium on Imprecise Probabilities and Their Applications*, Bernard, Seidenfeld, and Zaffalon, Eds., Lugano, Switzerland, 2003, pp. 452–466.

[5] Brian D. Davison, *The design and evaluation of web prefetching and caching techniques*, Ph.D. thesis, Rutgers University, 2002.