UNIVERSITÉ DE NAMUR

FACULTÉ D'INFORMATIQUE

Grandgousier

Bot conseiller en vins (Prolog)

IA & programmation symbolique — IHDCM036

Gilles Homez - Mathieu Crotteux

Professeur : Jean-Marie Jacquet

Plan

Objectif

- Bot conversationnel en PROLOG pour conseiller des vins.
- Deux fonctions principales :
  - recommander (appellation, prix, plats) ;
  - expliquer (nez, bouche, description, appellation).
- Approche symbolique : mots-cles ponderes + regles + motifs.

Architecture generale

- Deux modules :
  1. base de connaissances (faits) ;
  2. moteur de dialogue (normalisation, regles, reponse).
- Chaine : `read_atomics/1` → `produire_reponse/2` → `ecrire_reponse/1`.

Base de connaissances : representation

- Identifiant stable par vin (atom Prolog).
- Faits : `nom/2`, `prix/2`, `provenance/2`, `appellation/2`.
- Textes : `nez/2`, `bouche/2`, `description/2` (listes de lignes).

```
nom(chambolle_musigny_premier_cru_2012,
 'Chambolle Musigny 1er Cru 2012 - Les Noirots').
prix(chambolle_musigny_premier_cru_2012, 63.85).
appellation(chambolle_musigny_premier_cru_2012, chambolle_musigny).
provenance(chambolle_musigny_premier_cru_2012, bourgogne).
```

Normalisation de l'entree

- Objectif : rendre le matching robuste aux variantes d'ecriture.
- Etapes :
  - unifier les noms de vins (long/court/compact) ;
  - normaliser les appellations et les plats ;
  - decomposer certains tokens composes.

```
normaliser_question(Lin,Lout) :-
  nom_vins_uniforme(Lin,L1),
  normaliser_appellations_tokens(L1,L2),
  normaliser_plats_tokens(L2,L3),
  expand_compound_tokens(L3,L4),
  maplist(normaliser_mot,L4,Lout).
```

Mots-cles ponderes et regles

- `mclef/2` priorise l'intention (nez/bouche > vin/vins).
- Une regle : `regle_rep(MotCle, Id, Pattern, Rep) :- Conditions.`
- Matching : sous-liste dans la question normalisee.

```
mclef(nez,10).
mclef(bouche,10).
mclef(appellation,8).
mclef(vins,5).

regle_rep(nez,1,[quel,nez,presente,le,Vin],Rep) :-
  nez(Vin,Rep), memoriser_vin(Vin).
```

Contexte minimal : dernier vin

- Gere les questions elliptiques (ex. "puis-je le boire maintenant").
- Memoire : `dernier_vin/1` mis a jour quand un vin est cite.

```
:- dynamic dernier_vin/1.

memoriser_vin(Vin) :-
  retractall(dernier_vin(_)),
  asserta(dernier_vin(Vin)).
```

Validation : tests automatises (plunit)

- Suite `plunit` :
  - base (provenance/appellation pour chaque vin) ;
  - normalisation (formes compactes, variantes) ;
  - reponses (Bourgogne, prix, nez/bouche, plats, cas vides).

```
test(all_wines_have_taxonomy) :-
  findall(V, nom(V,_), Vins),
  forall(member(V,Vins),
         (provenance(V,_), appellation(V,_))).
```

Limites et perspectives

- Limites :
  - couverture linguistique dependante des patterns ;
  - ambiguite si plusieurs mots-cles forts ;
  - heuristiques sensibles au lexique.
- Pistes :
  - enrichir synonymes et patrons ;
  - memoriser aussi le dernier filtre (appellation/prix) ;
  - scoring global avant choix final.

Conclusion

- Base riche + moteur de regles robuste.
- Normalisation : comprehension fiable sans NLP complexe.
- Tests `plunit` : fiabilite et non-regression.

Merci !