Computer simulation of word associations and crossword solving

Sigurd, Bengt

1996

Document Version:
Other version

Lund University, Dept. of Linguistics
Working Papers 45 (1996), 157–166

Computer simulation of word associations and crossword solving

Bengt Sigurd

When you say *house* to somebody, he or she might associate to *mouse* or to *mortgage*. People's associations to words follow two main paths: form or meaning – sometimes both. Associations to form may be associations to the written form (spelling) or to the pronunciation (or to the pronunciation derived from the spelling). The spelling might influence the associations. If you hear the word *bill* you might associate it to *money*, but if you have seen the spelling *Bill* the association *Clinton* is more natural, at least to an American these days.

There has been much research concerning associations and suggestions that some associations are more normal than others (see e.g. Hörmann 1979). When given the word *Tisch (table, table*, respectively) 29% of the German subjects first answered *Stuhl*, 53 of the French subjects *chaise* and 84% of the English subjects *chair*. When presented with the word for *hand*, 20% of the German subjects, 25% of the French subjects and 26 of the English subjects replied with the word for *foot*. The response to the word for *soft* was in about 40% of the cases words for *hard*, the response to words for *eating* was in about 25% of the cases words for *drink*. The word *Haus* elicited *Hof* from 14% of the German subjects. *Maison* was answered by *toit* by 14% of the French subjects and *house* was answered by *home* by 25% of the English subjects.

Psychologists have long taken an interest in the free associations and lists of associations. Some experiments have also measured the reaction times. Free associations interested Freud as unexpected associations or avoided associations could have sexual reasons according to him. Psychologists have noted that children have more syntagmatic associations, i.e. associations to words which often follow, as e.g. *dog : barks, fine : weather*. Adults tend to have more paradigmatic associations, i.e. associations to words of the same grammatical category, as illustrated by *cat* when the stimulus is *dog*, and *run* when the stimulus is *walk*.

Some associations are fairly general but others are most individual. If I hear the word *chestnut* I think of our cocker-spaniel not the tree or nut as I guess most people would do. Memories and experience play an important role in the associations. When talking to somebody one sometimes wonders how he or she got from a certain topic to another topic. The steps in the train of thoughts are not always mentioned overtly or clear. The individual nets of associations and the mutual nets constitute interesting fields of research. Knowledge of these nets and the possibility of simulating the associations by computer may have practical applications, e.g. for marketing (trade and product names), psychological diagnosing, forensic investigations, cryptology – and crossword solving.

This note presents some simple Prolog programs which can be used to simulate form and meaning associations, chains of associations and crossword solving processes. Related semantic programs have been worked out by others. Quillian 1968 is an early attempt to simulate certain memory processes. The project *WordNet* by Miller et al. 1993 is an advanced lexical database constructed to allow searching for synonyms, antonyms, hypernyms and whole-part relations. The programs and lexicons to be discussed in this paper are only made for experimental and exploratory purposes. The programming language used is LPA MacProlog32. I think Prolog is a convenient language for expressing the ideas of the paper, but I do not maintain that Prolog is the best computer program for the purpose. A certain knowledge of Prolog programming is an advantage for the reader.

Phonetic associations

The phonetic associations rely on phonetic similarity, which may be of different kind and size. The word *hill* is clearly phonetically similar and may be a natural association to *kill* which is explained by the fact that only the first letters (sounds, segments) differ. Also the word *kiss* has a phonetic resemblance to *kill* but seems less natural as the difference is in the last sound (letters). Words which rhyme are easy to associate. The following Prolog command produces rhymes.

```
make_rhyme(Word,Rhyme) :-
    ccat(Cons,Rest,Word),ccat(Vow,Fin,Rest),vlist(L),member(Vow,L),
    (clist(L1),member(Cons1,L1),ccat(Cons1,Rest,Rhyme); % new cons
    Rhyme=Rest). % no consonant added
```

The rule changes the word in the variable *Word* into the rhyme *Rhyme*, e.g. *house* into *mouse*, *chase* into *case*, *sorrow* into *borrow*, *stride* into *wide*. The

rule is based on the fact that rhyming means identical from the stressed vowel and onwards. The initial consonant or cluster disappears and the rest is combined with a new consonant. The rule is based on spelling. It distinguishes the part beginning in a vowel in the first line and adds a consonant or nothing. The command *ccat(Cons,Rest,Word)* divides *Word* into *Cons* followed by *Rest*, and *ccat(Vow,Fin,Rest),vlist(L),member(V,L)* finds out whether *Rest* begins in a vowel by checking whether *Vow* is a member of the list *L*. If so *ccat(Cons1,Rest,Rhyme)* concatenates the element *Rest* after a new consonant *Cons1* producing *Rhyme*. The predicates *vlist(L)* and *clist(L1)* call lists of vowels and consonants, respectively. The lists used here are defined as follows:

```
vlist([a,e,i,y,o,u]).
clist([b,c,d,f,g,h,j,k,l,m,n,p,q,r,s,t,v,w,x,z]).
```

The rule above only adds single consonants unless clusters such as *st, str, spr, pr, pl* are included in the list of consonants. In poetry, a distinction is made between rhymes for the ear (proper rhymes) and rhymes for the eye. The rule cannot produce *aunt* as a rhyme to *plant* and suggests *grow* as a rhyme to *now* as it relies solely on the spelling. It does not take stress or tone accents into account and if the word includes several vowels it generates solutions for each vowel. If the rule is not restricted by requiring that the words generated should be found in a lexicon, it will produce a number of nonsense words as illustrated by the following printout of the result of the following command:

```
make_rhyme(generate,Rhyme)
```

Rhyme=benerate, cenerate, denerate, fenerate, jenerate, kenerate, lenerate, menerate, nenerate, penerate, qenerate, renerate, senerate, tenerate, venerate.

The following command generates some existing and some nonsense (or potential) words:

```
make_rhyme(space,Rhyme)
```

Rhyme=bace, cace, dace, face, gace, hace, jace, kase, lace, mace, nace, pace, qace, race, sace, tace, vace.

Rules which exchange the last vowel, the last consonant, the last two or three letters, etc. can be written in similar ways. A rule changing the last vowel can be written as follows:

```
changelastvowel(X,X1) :-
```

```
ccat(B,V,X),vlist(L),member(V,L),member(V1,L),ccat(B,V1,X1).
```

This rule will produce the following words if the command is *changelastvowel(cantina,X)*:

X=cantine, cantini, cantiny,cantino,cantinu.

The following rules change the *n*th vowel or consonant, respectively, changing the word *X* into *X1*.

```
nthvowelchange(N,X,X1) :- M is N-1,ccat(A,R,X),length(A,M1),
    M=M1,ccat(V,Rest,R),vlist(L),member(V,L),member(V1,L),
    V\=V1,ccat(V1,Rest,R1),ccat(A,R1,X1). % changes nth vowel

nthconschange(N,X,X1) :- M is N-1,ccat(A,R,X),length(A,M1),
    M=M1,ccat(V,Rest,R),clist(L),member(V,L),member(V1,L),
    ccat(V1,Rest,R1),V\=V1,ccat(A,R1,X1). % changes nth cons
```

If *nthvowelchange(5,cantina,X)* is applied we may get:

cantena,cantuna,cantyna, cantana, cantona

If the rules are to generate only existing words the result must be restricted by checking a lexicon. The following section presents a lexical format which can be used for such purposes.

Semantic associations

Semantic associations and chains of associations may be simulated using a lexicon including information about the word form, grammatical category, synonym, hypernym, antonyms and various other structured or unstructured associations. The following is such a lexicon to be used for experiments and demonstrations. It has the same basic format as the lexicon used in Swetra Referent Grammar (Sigurd 1994). The first slot contains the word form. The second slot contains the grammatical category. The third slot includes a synonym; if no synonym is found, the same word is given. The fourth slot includes a hypernym. The fifth slot includes an unstructured list of words of all categories, which the author has associated to. The lexicon is not claimed to have any generality. More specific slots may be included, e.g. one special slot for typical object and subject with a verb, typical head of an adjective, etc.

English lexicon

```
elex(dog,n,dog,animal,[wolf,cat,angry,bark,bite]).
elex(costumer,n,consumer,human,[merchandise,price,shop,buy,
```

```
    economic]).
elex(hand,n,hand,bodypart,[foot,take,finger,greet]).
elex(merchandise,n,goods,goods,[customer,buy,shop,store]).
elex(shop,n,store,company,[money,dealer,price,open,merchandise]).
elex(finger,n,finger,bodypart,[nail,thumb,dirty]).
elex(dirty,a,dirty,property,[dirt,dig,mud]).
elex(shrewd,a,cunny,property,[cat,owl,fox,smart]).
elex(angry,a,furious,property,[dog,furious,evil,strike,boss]).
elex(cat,n,cat,animal,[rat,mouse,dog,chase,purr,shrewd,miew]).
elex(fox,n,fox,animal,[shrewd,take,chase,hunt]).
elex(rat,n,mouse,animal,[cheese,trap,chase,mouse,cat]).
elex(purr,v,purr,sound,[cat,calm,sleep,sound]).
elex(peep,v,peep,sound,[rat,mouse,sound,wheene,suffer]).
elex(chase,v,hunt,move,[dog,cat,prey,mouse,rat]).
elex(wolf,n,wolf,animal,[howl,chase,hurt,dangerous]).
elex(chese,n,chese,food,[food,eat,milk,cream]).
elex(food,n,food,food,[eat,good,mouth,knife,fork,spoon]).
elex(dangerous,a,threatening,property,[evil,hurt,strike]).
elex(bark,v,bark,sound,[dog,angry,howl]).
elex(howl,v,bark,sound,[wolf,dog,angry,night]).
elex(bite,v,bite,activity,[dog,chew,trousers]).
elex(goods,n,goods,goods,[price,shop,buy,store]).
elex(foot,n,foot,bodypart,[hand,toe,walk,kick]).
elex(take,v,grip,activity,[hand,grab,steal,grip]).
elex(greet,v,salute,activity,[hand,cap,meet,hallo,friend]).
elex(house,a,building,building,[build,mortgage,ground,garden,chimney]).
elex(mouse,n,rat,animal,[cheese,trap,cat,chase,rat]).
```

The following rule renders (near) synonyms, e.g. *furious* to *angry*. The first line gives only synonyms which are in the lexicon (*elex*).

```
esynonym(X,X1) :- elex(X,_,X1,_,_),elex(X1,_,_,_,_),print(X1),nl.
esynonym(X,X1) :- elex(X,_,X1,_,_),print(X1),nl.
```

The following rule renders verb associations (found in the current lexicon), e.g. *bark* or *bite* to *dog*

```
everb(X,X1) :-  elex(X,_,_,_,L),member(X1,L),elex(X1,v,_,_,_),print(X1),nl.
```

The following rule gives an associated adjective, e.g. *shrewd* to *fox.*

```
eadj(X,X1) :-     elex(X,_,_,_,L),member(X1,L),elex(X1,a,_,_,_),print(X1),nl.
```

The following rule returns a noun, e.g. *trap* to *mouse.*

```
enoun(X,X1) :-  elex(X,_,_,_,L),member(X1,L),elex(X1,n,_,_,_),print(X1),nl.
```

The rules may return several words (solutions) if the list of free association in the last slot includes several words of the type required.

The following rule is basic to the simulation of associations and chains of associations. It picks words from the list of loose associations in the last slot.

```
eassociation(X,X1) :- elex(X,_,_,_,L),member(X1,L),print(X1),nl.
```

Using a series of association commands, chains of associated words are generated as illustrated below. The primary stimulus *X* is set at *dog*. The conditions *X* to be different from *Y, X* to be different from *Z*, etc. are used in order to avoid repetition (recursion) of the same words. The command has several solutions for the lexicon used, some of which are shown.

```
X=dog,eassociation(X, Y), eassociation(Y, Z), eassociation(Z, W), X\=Z,
    X\=W, Y\=W, nl
```

Solutions:

No.1 : X = dog, Y = wolf, Z = howl, W = angry
No.2 : X = dog, Y = wolf, Z = howl, W = night
No.3 : X = dog, Y = wolf, Z = chase, W = cat
No.4 : X = dog, Y = wolf, Z = chase, W = prey
No.5 : X = dog, Y = wolf, Z = chase, W = mouse
No.6 : X = dog, Y = wolf, Z = chase, W = rat
No.7 : X = dog, Y = wolf, Z = dangerous, W = evil
No.8 : X = dog, Y = wolf, Z = dangerous, W = hurt
No.9 : X = dog, Y = wolf, Z = dangerous, W = strike
No.10 : X = dog, Y = cat, Z = rat, W = chese
No.11 : X = dog, Y = cat, Z = rat, W = trap
No.12 : X = dog, Y = cat, Z = rat, W = chase
No.13 : X = dog, Y = cat, Z = chase, W = prey
No.14 : X = dog, Y = cat, Z = chase, W = mouse
No.15 : X = dog, Y = cat, Z = chase, W = rat
No.16 : X = dog, Y = cat, Z = purr, W = calm
No.17 : X = dog, Y = cat, Z = purr, W = sleep
No.18 : X = dog, Y = cat, Z = purr, W = sound
No.19 : X = dog, Y = cat, Z = shrewd, W = owl
No.20 : X = dog, Y = cat, Z = shrewd, W = fox
No.21 : X = dog, Y = cat, Z = shrewd, W = smart
No.22 : X = dog, Y = bark, Z = angry, W = accuse

All words in the lexicon do not allow such long chains. Some words are included in more association lists. Not all the words in the free association list are lexical words (*elex* entries) and only the last word (variable) may be a word which is not in the lexicon. The following are examples of other runs in this lexicon.

```
X=hand, eassociation(X, X1), eassociation(X1, X2), eassociation(X2,
    X3), X\=X2, X\=X3, X1\=X3
No.1 : X = hand, X1 = finger, X2 = dirty, X3 = dirt

X=greet
No.1 : X = greet, X1 = hand, X2 = foot, X3 = toe

X=fox
No.1 : X = fox, X1 = shrewd, X2 = cat, X3 = rat
```

One may also ask for the chain between a certain final word and the first (stimulus) word using the following command if a chain of three words are required. The following command asks for the steps between *dog* and *cheese*.

```
X=dog,X3=cheese,eassociation(X,X1),eassociation(X1,X2),
    eassociation(X2, X3), X\=X2, X\=X3, X1\=X3
```

The following is a similar experimental Swedish lexicon. Given the English lexicon it should not be difficult to understand the Swedish words.

```
lex(hund,n,hund,djur,[varg,katt,arg,skälla,bita]).
lex(kund,n,konsument,human,[konsument,varor,priser,affär,köpa,
    sparsam]).
lex(hand,n,hand,kroppsdel,[fot,ta,finger,hälsa]).
lex(varor,n,gods,gods,[kund,köpa,affär]).
lex(affär,n,butik,hus,[pengar,handla,öppen,varor,butik]).
lex(finger,n,finger,kroppsdel,[nagel,tumme,smutsig]).
lex(smutsig,a,skitig,egenskap,[smuts,skit,jord]).
lex(listig,a,klok,egenskap,[katt,uggla,räv]).
lex(arg,a,ilsken,egenskap,[bov,varg]).
lex(katt,n,katt,djur,[råtta,mus,hund,jaga,spinna,listig]).
lex(räv,n,mickel,djur,[listig,ta,jaga]).
lex(råtta,n,mus,djur,[ost,pipa,jaga]).
lex(spinna,v,surra,låta,[katt,lugn,låta]).
lex(pipa,v,pipa,låta,[råtta,mus,gråta]).
lex(jaga,v,förfölja,förflytta,[hund,katt,råtta,byte]).
lex(varg,n,varg,djur,[yla,jaga,riva,farlig]).
lex(ost,n,ost,mat,[mat,äta,mjölk]).
lex(mat,n,föda,livsmedel,[äta,gott,mun,kniv,gaffel,sked]).
lex(arg,a,rasande,levande,[ond,slå,klaga,skälla]).
lex(farlig,a,hotfull,levande,[ond,skada,slå]).
lex(skälla,v,skälla,ljud,[hund,arg]).
lex(yla,v,skälla,ljud,[hund,arg,varg]).
lex(bita,v,bita,djur,[hund,byxor,tugga]).
lex(varor,n,gods,ting,[priser,affär,köpa,lager,lagra]).
lex(fot,n,fot,kroppsdel,[hand,tår,gå,sparka]).
lex(ta,v,gripa,levande,[hand,handtag,grepp]).
lex(hälsa,v,hälsa,levande,[hand,mössa,möte,vän]).
```

Using the Swedish lexicon the following are some of the long chains of association which can be generated.

```
X=hund,association(X,X1),association(X1,X2),association(X2,X3),
    association(X3,X4),association(X4,X5),X\=X2,X\=X3,X\=X4,X\=X5,
    X1\=X3,X1\=X4,X1\=X5,X2\=X4,X2\=X5,X3\=X5
```

No.1 : X = hund, X1 = varg, X2 = jaga, X3 = katt, X4 = råtta, X5 = ost
No.2 : X = hund, X1 = varg, X2 = jaga, X3 = katt, X4 = råtta, X5 = pipa
No.3 : X = hund, X1 = varg, X2 = jaga, X3 = katt, X4 = spinna, X5 = lugn
No.9 : X = hund, X1 = varg, X2 = jaga, X3 = råtta, X4 = ost, X5 = mjölk
No.10 : X = hund, X1 = varg, X2 = jaga, X3 = råtta, X4 = pipa, X5 = mus
No.14 : X = hund, X1 = katt, X2 = råtta, X3 = ost, X4 = mat, X5 = mun
No.15 : X = hund, X1 = katt, X2 = råtta, X3 = ost, X4 = mat, X5 = kniv

It is also possible to connect semantic and phonetic associations. Such resulting chains are illustrated by the series:

rat: mouse: house
singer: finger: nail: sail

Crossword solving?

Our experimental system has used a restricted lexicon to show how word associations can be simulated. A human being has, of course, an enormous lexicon of associations which can hardly be imitated in full by computer systems. Existing synonym lexicons and thesauruses only include some of the word associations and are of restricted use. If the associations of one individual person or a group of persons, e.g. a certain family or a profession, are to be simulated, the lexicon has to be built according to empirically observed associations.

The ideas presented here can also be applied in the construction of an automatic crossword solver (and even an automatic crossword constructor). The form information given for the requested word is its number of letters and (cross) letters which also occur in other words. For each word filled, in a number of letters in other words will be known. The semantic information may be a synonym, a negated hypernym, a typical subject or object or a more or less loose association, natural in simple crosswords, far-fetched in advanced cases. Thus the word sought may for instance include five letters and the cue may be: *likes cheese*. In our small lexicon the word *cheese* is found in the list of free associations of both *mouse* and *rat*, but only *mouse* has five letters.

The following is a command which returns an associated English word *X* of length *L* given the cue *Cue*.

```
crossword(L,Cue,X) :- eassociation(Cue,X),length(X,L1),L1=L.
```

The problem in crosswords is generally also the requirement that the letter in a given position should be certain letter. The following command checks the letter *Letter* in position *N* in the word *Word*. If one also wants the letter to be a vowel the command *vlist(V),member(B,V)* has to be added.

```
checknthletter(N,Word,Letter) :-
    ccat(In,Rest,Word),
    length(In,M),M is N-1,ccat(B,Fin,Rest),length(B,L),L=1,Letter=B.
```

The problem is often to find a word (*W1*) of e.g. five letters beginning in the same letter as another word (*W2*) of six letters, which should end in a letter which is the second letter of a third word (*W3*) of seven letters. If we define a command *c(Length,Word,Position,Letter)* along the lines mentioned before we can set up the following series of commands in order to arrive at the words *W1*, *W2*, *W3* (and the letters *Letter1* and *Letter2*) for this case:

```
c(5,W1,1,Letter1),c(6,W2,1,Letter1),
c(6,W2,6,Letter2),c(7,W3,2,Letter2)
```

This kind of equation system can be solved by Prolog by searching a lexicon. The following words e.g. fit: *W1 = style*, *W2 = string*,*W3 = agitate*. If the words to appear in the crossword are given (but not the semantic cues) as in some types of crosswords one may often arrive at only one solution. In normal types of crosswords, however, a lexicon such as the *Word list of the Swedish Academy* (*SAOL*) has to be consulted and there will generally be several solutions.

But the design of an automatic crossword solver also using semantic associations requires a large sophisticated lexicon based on inclusive studies of the typical crossword associations and we will not discuss this project further here. The usefulness of this project, however, seems dubious, as – if successful – it would take away the pleasure of solving crossword puzzles alone or in company.

References

Hörmann, Hans. 1979. *Psycholinguistics*. New York: Springer-Verlag.

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross & Katherine Miller. 1993. *Introduction to WordNet: An on-line lexical database* (mimeographed).

Quillian, M. R. 1968. 'Semantic Memory'. In M. Minsky. (ed.), *Semantic information processing.* Cambridge, Mass.: MIT Press.

Sigurd, Bengt. 1994. *Computerized grammars for analysis and machine translation*. Lund: Lund University Press.