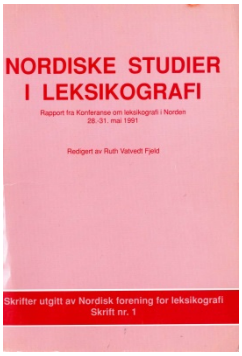
NORDISKE STUDIER I LEKSIKOGRAFI

Titel: Automatisk analyse av maskinleselige ordbøker til bruk i en orddatabase

Forfatter: Jostein Baustad

Kilde: Nordiske Studier i Leksikografi 1, 1992, s. 423-431
Rapport fra Konferanse om leksikografi i Norden, 28.-31. mai 1991

URL: http://ojs.statsbiblioteket.dk/index.php/nsil/issue/archive



Søgbarhed
Artiklerne i de ældre Nordiske studier i leksikografi (1-5) er skannet og OCR-behandlet. OCR står for ’optical character recognition’ og kan ved tegngenkendelse konvertere et billede til tekst. Dermed kan man søge i teksten. Imidlertid kan der opstå fejl i tegngenkendelsen, og når man søger på fx navne, skal man være forberedt på at søgningen ikke er 100 % pålidelig.

Jostein Baustad

Automatisk analyse av maskinleselige ordbøker til bruk i en orddatabase

Denne artikkelen skisserer hvordan automatisk analyse av maskinleselige ordbøker til bruk i en orddatabase kan foregå. To typer orddatabaser med ulik detaljeringsgrad beskrives, og to ganske forskjellige framgangsmåter skisseres. Den ene typen orddatabase egner seg godt for vanlige ordboksoppslag, det vil si ved at brukeren oppgir et oppslagsord og får presentert informasjon om oppslagsordet. Den andre typen orddatabase egner seg for avanserte databasesøk blant ordene i databasen. Et eksempel på et slikt søk kan være "finn alle substantiv som stammer fra italiensk og har med domenet musikk å gjøre".

1 Mål og hensikt

Denne artikkelen skisserer en framgangsmåte som er benyttet for å legge *Bokmålsordboka* fra Universitetsforlaget inn i en database. Tilsvarende framgangsmåte eller deler av den er også benyttet for andre ordbøker, blant disse *Norsk-Engelsk ordbok* og *Fremmedordboka* fra Kunnskapsforlaget.

Ei ordbok i en database kan brukes på mange måter. To av disse er interessante i denne sammenhengen :

* Ordboka kan brukes som ei tradisjonell ordbok, det vil si ved at brukeren oppgir et oppslagsord og får presentert all informasjon om dette ordet som ligger i basen. Dersom brukeren er inne i for eksempel et tekstbehandlingsprogram, kan han ved hjelp av et tastetrykk slå opp det ordet som markøren befinner seg i. Vi bruker et program kalt *WordSmith* til denne typen oppslag.

* Ordboka kan brukes til å utføre avanserte databasesøk, det vil si finne alle oppslagsord i ordboka som tilfredsstiller de kriterier brukeren oppgir. Et eksempel kan være å finne alle intetkjønnsord som stammer fra italiensk, et annet eksempel kan være å finne alle ord som har med musikk å gjøre. Til denne typen søk bruker vi et program kalt *Lexical Query Language* (LQL).

For å kunne bruke et av programmene ovenfor (eller andre programmer eller verktøy for den saks skyld) er det en forutsetning at ordboka er i et format som programmet kjenner. Det vil si at WordSmith og LQL trenger ulik kunnskap om ei ordbok. Mens WordSmith bare trenger å vite oppslagsordet samt hva som skal presenteres når dette ordet slås opp, så trenger LQL detaljert kunnskap om ordklasse, ordhistorie, domener, definisjoner og så videre. WordSmith benytter derfor et format kalt *Dictionary Access Method* (DAM), mens LQL arbeider med ordbøker i *Lexical Data Base-format* (LDB). Disse formatene beskrives nærmere seinere.

Når ei ordbok skal legges inn i en database, er det hovedsaklig to måter det kan gjøres på. Den ene måten er å bygge opp ordboka manuelt fra grunnen av. Dette er selvsagt veldig tidkrevende, men en kan selv bestemme hvilke opplysninger som en ønsker å ha med i ordboka. Den andre måten baserer seg på mest mulig automatikk ved at en lager et program som leser og analyserer datafiler fra ordboka, og så legger inn hvert oppslagsord med tilhørende opplysninger i databasen. Denne måten krever at man har tilgang til datafiler fra ordboka. Slike filer kan vanligvis skaffes fra forlaget som har utgitt ordboka ved at man kjøper rettigheter til å bruke den i maskinleselig utgave. Det er denne siste måten som beskrives i denne artikkelen.

Datafilene fra forlaget, heretter kalt kildefiler, er vanligvis ikke i det formatet man ønsker. Derfor må kildefilene konverteres til et format som gjør at ordboka kan brukes slik en vil. Det er naturligvis meget viktig å vite hva en ønsker å bruke ordboka til før en utfører konverteringen. Hvis ordboka kun skal brukes som ei tradisjonell ordbok, er det unødvendig å utføre den krevende konverteringen til LDB-format, men dersom en ønsker å gjøre avanserte databasesøk i ordboka, er ikke DAM-formatet tilstrekkelig.

2 Ordboksformatene

2.1 Kildefiler

Kildefilene fra forlaget er vanligvis en av to typer. Ordbøkene er ofte tilgjengelig i form av trykksatsfiler, med andre ord filer som er beregnet på den maskinen som skal trykke ordboka. Vanligvis er slike filer en eneste lang sekvens av tegn og koder. Tegnene er naturligvis de tegnene som kommer fram i trykk, mens kodene er koder som bestemmer utseendet og posisjonene til tegnene, for eksempel koder for normal, uthevet eller kursiv skrift, koder for linjeskift, innrykk eller ny artikkel. Kodene kan også representere tegn som ikke finnes i det aktuelle tegnsettet, for eksempel matematiske tegn som integraltegn eller kvadratrottegn eller greske tegn som alfa og omega.

Figur 1 nedenfor viser et eksempel på hvordan en trykksatsfil kan se ut, og hvordan den korresponderende delen av ordboka ser ut. Eksempelet er hentet fra *Fremmedordboka*, og oppslagsordet er *abacist* med litt omkringliggende tekst.

```
fra¤¤<->Hamburg.<QL>¤¤<UF2>abaci<ek66>st, <1>en, g
r., regnemester. - <3>a<ek66>bacus, <1>en, se¤¤<->
abakus.<QL>¤¤<UF2>abaddo<ek66>n, <1>hebr., eg. øde


ab, prep., lat., av, fra, f.eks. ab Hamburg, fra
  Hamburg.
abaci'st, en, gr., regnemester. - a'bacus, en, se
  abakus.
abaddo'n, hebr., eg. ødeleggelse, undergang, i
  Det gamle testamente bet. for dødsriket.
```

Figur 1 - Eksempel på trykksatsfil og hvordan den ser ut i ordboka.

Som vi ser er nesten alle kodene avgrenset av mindre-enn og større-enn-tegn, "<" og ">". Vi ser at noen koder er koder for tekstattributter som uthevet, "<3>" eller kursiv, "<2>", mens andre koder representerer spesialtegn, som trykkaksenten ', "<ek66>". "Bomba" ("¤") står her for tegn som ikke kan vises på trykk, i dette eksempelet tilsvarer to "bomber" et linjeskift.

Den andre typen kildefiler er på mange måter lik den første typen, men slike filer har en klarere struktur og ofte inneholder de opplysninger som ikke direkte kommer fram i den trykte utgaven av ordboka. Filene er gitt en klarere struktur ved for eksempel å ha linjeskift (i stedet for koder for linjeskift) der det er linjeskift i boka og gjerne litt "luft" mellom artiklene. Men den viktigste forskjellen fra trykksatsfiler er at de strukturte filene ofte inneholder feltkoder som ikke kommer fram i den trykte ordboka. Feltkodene er koder som forteller hvilken type opplysning som følger etter koden, for eksempel ordhistorie, definisjon eller uttrykk der oppslagsordet forekommer.

Figur 2 nedenfor viser et eksempel på hvordan en strukturert kildefil med feltkoder kan se ut, og hvordan den korresponderende delen av ordboka ser ut. Eksempelet er hentet fra *Bokmålsordboka*, og oppslagsordet er *fell* med litt omkringliggende tekst.

```
=
NB001      fell
NB001a     M1
TR007
..OPP      ##$Cfell@ m1
..ETY      (norr $Bfeldr@)
..DEF      $C1@ dyrepels
..DEF      $C2@ skinn til overbredsel
..UTR      $Bkrype under f-en@
=
NB001      fellah
NB001a     M1
TR007
..OPP      ##$Cfellah@ m1
..ETY      (fra arab 'bonde')
..DEF      egyptisk bonde


fell m1 (norr feldr) 1 dyrepels 2 skinn til
  overbredsel krype under f-en
fellah m1 (fra arab 'bonde') egyptisk bonde
```

Figur 2 - Eksempel på strukturert kildefil med feltkoder og utseendet i ordboka.

Vi ser at alle feltkodene finnes helt til venstre på hver linje. De fleste feltkodene er ganske intuitive, men en forklaring på noen av kodene følger. "NB001" er en kode som kommer foran selve oppslagsordet. Her er oppslagsordet uten eventuelle stammestreker eller trykkaksenter. Dersom flere oppslagsord deler samme artikkel, som for eksempel *felgbrems* og *felgbremse*, vil vi ha ett oppslagsord på en "NB001"-linje og det andre på en "NB002"-linje. Feltkoden "..ETY" kommer foran opplysninger om etymologi (ordhistorie), mens "..DEF" kommer foran selve definisjonene. Som vi ser, kommer ulike definisjoner på separate "..DEF"-linjer. Fra eksempelet kan vi dessuten se at det også for denne ordboka er koder for

skriftattributter, her "$B" for kursiv og "$C" for uthevet. Denne skrifttypen oppheves av en "@"-kode.

2.2 DAM-formatet

Dictionary Access Method (DAM) er et databasesystem spesielt beregnet for ordbøker. DAM lagrer ordbøkene i et format som også kalles DAM. Det som kjennetegner DAM-formatet, er at hver post i databasen kun består av to felter, et felt for selve oppslagsordet og et informasjonsfelt for informasjonen om oppslagsordet. Hva informasjonsfeltet inneholder, vil avhenge av hvilken type ordbok det er. En synonymordbok vil ha ulike synonymer til oppslagsordet, mens en tospråklig ordbok vil ha ulike oversettelser av oppslagsordet. I begge tilfeller kan informasjonsfeltet inneholde opplysninger om ordklasse, bøyning og lignende. Men uansett hvor mye eller lite som ligger i informasjonsposten, så vet ikke DAM-systemet noe om hva de forskjellige opplysningene er, siden alle opplysningene ligger i det samme informasjonsfeltet. Som navnet antyder arbeider DAM-systemet med dataene i databasen på samme måte som en vil bruke ei trykt ordbok, det vil si via oppslagsordet. DAM-systemet tilbyr alle operasjoner en kan forvente av et databasesystem, som innsetting, sletting og oppdatering av poster, samt ulike måter å hente informasjon fra databasen på, for eksempel ved hjelp av oppslagsord eller ordnummer. DAM tilbyr også mulighet for at brukeren selv kan bestemme sorteringsrekkefølgen for oppslagsordene i ei ordbok. Det betyr at DAM kan benyttes både for filer i ASCII og EBCDIC-format, og vi kan for en gangs skyld få våre "eksotiske" 3 tegn æ, ø og å sortert i riktig rekkefølge! I tillegg er det mulig å be om at blanke, bindestreker og lignende skal overses ved sortering, slik at *ad hoc* sorteres som *adhoc*, eller for eksempel at tall skal sorteres som om de var skrevet med bokstaver, slik at *A4* sorteres som *Afire*.

2.3 LDB-formatet

En klar begrensning ved DAM-formatet er at opplysningene i informasjonsfeltet ikke er skilt fra hverandre. *Lexical Data Base* (LDB) er et ordboksformat som ikke har denne begrensningen. LDB har alle fordelene som DAM-formatet har og kan i tillegg skille de ulike opplysningene om oppslagsordet. Hver post i databasen har et vilkårlig antall felter hvor de ulike opplysningene plasseres, for eksempel felt for homografnummer, ordklasse og selve definisjonen. Feltene har en hierarkisk struktur som minner om en trestruktur. Denne strukturen svarer til brukerens oppfatning om organiseringen av de ulike opplysningene i en ordartikkel.

Figur 3 nedenfor viser hvordan en post i LDB-formatet ser ut (logisk). Oppslagsordet er *fell* fra *Bokmålsordboka*, og hvordan dette ser ut i den trykte utgaven av ordboka kan finnes i figur 2. Som vi ser, består treet av to typer greiner, terminerende og ikke-terminerende. Terminerende greiner representerer de feltene i databasen som inneholder de ulike opplysningene om et ord. "ordklasse" og "defnummer" er eksempler på slike greiner. De ikke-terminerende greinene er de som skaper den hierarkiske strukturen. Disse greinene kan ikke inneholde ordopplysninger, men inneholder i stedet andre greiner. "homografnode" og "defnode" er eksempler på ikke-terminerende greiner. Vi ser blant annet at opplysninger som tilhører samme definisjon, er plassert under samme "defnode". Vi ser også at en del opplysninger som ikke står eksplisitt i ordboka, er utledet og finnes i treet. Dette gjelder for

eksempel ordklassen substantiv, som her er utledet fra bøyningskoden. Dessuten er den komprimerte formen *f-en* i uttrykket blitt utledet til *fellen*.

```
-entry
  -homografnode
    -oppnode
      -oppslagsord : fell
      -skriveform : hovedform
      -representasjon : fell
      -ordklasse : substantiv
      -boyningsnode
        -genus : m
        -boyningskode : m1
    -ordhistnode
      -spraak : norr
      -ord : feldr
    -defnode
      -defnummer : 1
      -definisjon : dyrepels
    -defnode
      -defnummer : 2
      -definisjon : skinn til overbredsel
      -uttrykksnode
        -uttrykk : krype under fellen
```

Figur 3 - Eksempel på den hierarkiske strukturen i en LDB-fil.

3 Konvertering til DAM-format

For å konvertere ei ordbok til DAM-format skal det altså lages et program som leser kildefilene, analyserer disse og legger de riktige opplysningene inn i databasen. Det finnes grensesnitt til DAM-systemet i flere ulike programmeringsspråk, blant annet Prolog og REXX, som er et Pascal-lignende språk. REXX er det språket som er benyttet i den framgangsmetoden som er beskrevet her. Programmets oppgave er å finne oppslagsordet med tilhørende informasjon fra kildefilene.

Hvor vanskelig denne oppgaven er, avhenger svært av hvordan kildefilene ser ut. For strukturerte kildefiler er det vanligvis ikke altfor vanskelig å kunne skille oppslagsord og informasjon. Kildefilene for *Bokmålsordboka*, jamfør figur 2, har oppslagsordene skrevet i korrekt skriveform på egne linjer ("NB"-linjer). Med korrekt skriveform menes her uten eventuelle stammestreker "|" og trykkaksenter "'" og alle tilder "~" er erstattet med den korrekte stammen. Det er derfor vanligvis en relativt enkel oppgave å finne oppslagsordet for slike kildefiler. Siden det kan være flere oppslagsord for hver artikkel, som for eksempel *felgbrems* og *felgbremse*, må alle oppslagsordene tas vare på til hele ordartikkelen er lest inn. De neste linjene settes sammen for å lage selve informasjonsfeltet. Det kan lønne seg å ta vare på alle feltkoder og alle skrifttypekoder i disse linjene. Feltkodene er svært nyttige dersom ordboka skal konverteres til LDB-format. WordSmith har mulighet for å undertrykke disse slik at feltkodene ikke vises når brukeren slår opp et ord. Når hele ordartikkelen er lest, legges oppslagsordene med tilhørende informasjonsfelt inn i databasen. Det er naturligvis

mulig å bearbeide informasjonsposten før ordet legges inn i basen. Dette er aktuelt ved for eksempel erstatning av komprimerte former, som i *krype under f-en*, som kan utledes til *krype under fellen*. Etter at ordet er lagt inn i databasen, er det klart for å lese neste ord, inntil hele kildefilen er lest.

For trykksatsfiler kan oppgaven å skille oppslagsord og informasjon om ordet være litt vanskeligere. I dette tilfellet må programmet selv finne start og slutt på hver ordartikkel. Siden hvert avsnitt i ordboka kan inneholde flere artikler, jamfør *abacist* og *abacus* i figur 1, må alt som står skrevet med uthevet skrifttype undersøkes. Dersom uthevet kun er benyttet for oppslagsordene, er dette relativt greitt. Men dersom uthevet også er benyttet til andre ting, for eksemepel definisjonsnummer eller inne i definisjonene, må programmet sjekke om det som står i uthevet er et nytt oppslagsord eller om det er en del av informasjonen om forrige oppslagsord. I enkelte tilfeller kan det være vanskelig for programmet å avgjøre dette, særlig i de tilfeller der uthevet er tillatt også inne i definisjonene. I tillegg til dette må programmet ta vare på stammer, slik at tilder "~" i seinere oppslagsord kan erstattes korrekt.

Figur 4 nedenfor viser et slikt eksempel. Tilden i *~brems(e)* må erstattes med den stammen som virker på dette stedet, *felg*. Dessuten må parentesene oppløses, slik at det blir to oppslagsord, *felgbrems* og *felgbremse*, med samme informasjon.

```
felg m1 (fra lty) hjulring som slange og dekk
  sitter rundt sykle på f-en / overf: være helt på
  f-en utslitt, nedbrutt ~brems(e) på sykkel:
  bremse på felgen, t forskj fra navbrems(e)
```

Figur 4 - Eksempel på ordartikkel med tilde og parenteser.

4 Konvertering til LDB-format

Konverteringen av ei ordbok til LDB-format foregår på en helt annen måte enn til DAM-format. Konverteringen utføres ved hjelp av en analysator kalt Dictionary Entry Parser (DEP). En forutsetning for å bruke DEP er at ordboka er i DAM-format. Oppgaven nå er å analysere informasjonsfeltet i DAM-postene for å finne de ulike opplysningene i dette feltet. Analysen foregår på grunnlag av et sett med grammatiske regler som sier hvordan opplysningene i ordboka er organisert (i forhold til hverandre). Ei liste over hvilke skilletegn eller "merker" (eng: token) som er brukt er også nødvendig. Denne lista inneholder alle tegnsekvenser som er benyttet i ordboka for å skille de ulike opplysningene fra hverandre. Lista vil derfor inneholde eventuelle feltkoder, koder for skrifttyper samt andre skilletegn som komma, semikolon og ulike typer parenteser. Disse skilletegnene gis symbolske navn.

Figur 5 nedenfor viser noen linjer fra lista benyttet for *Bokmålsordboka*. Vi kjenner igjen både skrifttypekoder og feltkoder fra tidligere eksempler. Det symbolske navnet lengst til høyre på hver linje kan velges fritt, men bør naturligvis ha ett fornuftig navn, slik som "font(normal)" for normal skrifttype.

De grammatiske reglene må ha en spesiell DEP-syntaks. Denne syntaksen ligger nær opptil Prolog-syntaks, med tillegg av en del spesialoperatorer. Figur 6 nedenfor viser noen grammatiske regler som finner definisjonsnummeret fra en definisjon i *Bokmålsordboka*.

```
delim("$A",font(normal)).
delim("$B",font(kursiv)).
delim("$C",font(uthevet)).
delim("@",fontend).

delim("^",linend).
delim("!!",entryend).

delim("TR007",entrybeg).
delim("..OPP",opp).
delim("..ETY",ety).
delim("..DEF",def).
delim("..UTR",utr).
```

Figur 5 - Eksempel på liste over skilletegn for DEP.

Definisjonsnummeret, "Defnr", er her sagt å være noe som ligger mellom skrifttypekoden for uthevet, "font(uthevet)", og koden som opphever denne skrifttypen, "fontend". Regelen "num_string" passer dessuten på at definisjonsnummeret kun består av tall. Reglene "num_string" og "defrec2" er ikke vist her, men må være definert et annet sted. "opt"-operatoren er en av DEPs tilleggsoperatorer. Den tillater argumentet i parenteser å være "optional", det vil si frivillig. Her betyr det at i noen tilfeller har vi et definisjonsnummer (når antall definisjoner er to eller flere), i andre tilfeller mangler dette nummeret (når det kun er en definisjon).

```
defrec     ==>  -def :
                opt(defnummer) :
                defrec2.

defnummer  ==>  -font(uthevet) :
                -Defnr :
                -fontend :
                num_string(Defnr).
```

Figur 6 - Eksempel på noen grammatiske regler for DEP.

Reglene virker på samme måte som Prolog-regler, enten så lykkes reglene, eller så mislykkes de. I kulissene blir de grammatiske reglene faktisk oversatt til Prolog-regler av analysatoren. Dersom de grammatiske reglene lykkes for et bestemt ord, betyr det at analysatoren har klart å analysere hele informasjonsfeltet og plassert opplysninger på greiner i treet i henhold til de grammatiske reglene. Dersom reglene mislykkes, kan dette ha flere årsaker. Den vanligste er at de grammatiske reglene ikke er omfattende nok til å ta hensyn til alle de ulike sammensetningene av koder og tekst. En annen årsak kan være at trestrukturen ikke er omfattende nok til å ta vare på alle ulike typer opplysninger i ordartikkelen. En tredje årsak som forekommer relativt sjelden, er direkte feil i kilden. Eksempel på slike feil kan være at uttrykk står i

normal skrift i stedet for kursiv, eller at forkortelsen for språket italiensk, "it.", mangler puntum ved en bestemt forekomst.

En av DEPs styrker er at de grammatiske reglene kan kombineres med vanlige Prolog-regler. Man kan derfor bruke vanlige Prolog-regler for å manipulere og kontrollere data før de legges inn i LDB-basen. Ofte er det nemlig slik at en tekststreng i en bestemt posisjon kan gi en av flere ulike typer opplysninger. Første opplysning etter selve oppslagsordet i Bokmålsordboka kan være en ordklasse, for eksempel "adj", eller en bøyningskode, for eksempel "m1". Disse to opplysningene skal inn på to forskjellige greiner, og reglene må derfor kontrollere at riktig opplysning legges inn på riktig plass.

Figur 7 nedenfor viser en helt vanlig Prolog-regel som avgjør om argumentet X er en av de kjente ordklassene. Denne regelen kan benyttes i de grammatiske reglene for å sikre korrekthet på opplysningene som legges inn i basen.

```
er_ordklasse(X)      <- Ordklasseliste=(
                     "adj"."adv"."art."."interj".
                     "konj"."pref"."prep"."pron".
                     "subst"."suff"."tallord"."v".
                     nil) &
member(X,Ordklasseliste).
```

Figur 7 - Eksempel på Prolog-regel som kan kombineres med grammatiske regler.

5 Avslutning

De fleste av dagens ordbøker er laget uten det formål for øyet at de i framtida skal kunne brukes i elektronisk form. Og dessverre er det noen av dem som vil lide litt av det når de forsøkes brukt i elektronisk form. Ei ordbok er som kjent et meget stort verk, og flere personer er vanligvis involvert i redigeringen. Det er derfor vanskelig å sørge for at hele ordboka er redigert på en konsekvent måte. Men viktigheten av dette kan ikke undervurderes dersom ordboka skal benyttes i elektronisk form. Mennesker har andre evner enn en datamaskin. I ei trykt ordbok kan et menneske forstå at en tekstsekvens er et uttrykk selv om den ikke står i kursiv skrift slik den burde ha gjort. Dette er vanskeligere å få en datamaskin til å forstå. Derfor er det veldig viktig at ordboka er redigert på en så konsekvent måte som over hodet mulig.

Litteratur

Berulfsen, Bjarne og Dag Gundersen. 1986. *Fremmedordbok.* 15. utg. Kunnskapsforlaget, Oslo.

Byrd, Roy. 1989. *LQL User Notes - An Informal Guide to the Lexical Query Language.* IBM Technical Report, Lexical Systems Project, I.B.M. Thomas J. Watson Research Center, Yorktown Heights, New York.

Byrd, Roy, Gustaf Neumann og Seved Andersson. 1988. *DAM - A Dictionary Access Method.* IBM Technical Report, Computer Science Department, I.B.M. Thomas J. Watson Research Center, Yorktown Heights, New York.

Kirkeby, W.A. 1983. *Norsk-Engelsk Ordbok.* Oslo.

Landrø, M. I., B. Wangensteen et al. 1986. *Bokmålsordboka.* Universitetsforlaget, Oslo.

Neff, Mary. 1988. *Using DAM: An introduction to the Dictionary Access Method.* IBM Technical Report, Lexical Systems Project, I.B.M. Thomas J. Watson Research Center, Yorktown Heights, New York.

Neff, Mary og Branimir Boguraev. 1991. *Dictionary Entry Parser Reference Manual.* IBM Technical Report, Lexical Systems Project, I.B.M. Thomas J. Watson Research Center, Yorktown Heights, New York.

Neff, Mary og Branimir Boguraev. 1991. *From Machine-Readable Dictionaries to Lexical Data Bases.* IBM Technical Report, Lexical Systems Project, I.B.M. Thomas J. Watson Research Center, Yorktown Heights, New York.

Neff, Mary og Roy Byrd. 1988. *WordSmith User's Guide: Version 2.0.* IBM Research Report RC13411, I.B.M. Thomas J. Watson Research Center, Yorktown Heights, New York.

Neff, Mary, Roy Byrd og Omneya Rizk. 1988. *Creating and Querying Hierarchical Lexical Data Bases.* Proceedings of the Second ACL Conference on Applied Natural Language Processing. 84-92. Austin, Texas.