**DEPARTMENT OF PHILOSOPHY,
LINGUISTICS AND THEORY OF SCIENCE**

Rekonstruktion av stavningsvarianter i SAOB

med normaliseringsregler från nusvenska till nysvenska

Charlotte Löfstrand

Masterarbete:	15 hp
Program:	Språkteknologi, masterprogram
Nivå:	Avancerad nivå
Termin och år:	Vår, 2023
Handledare:	Gerlof Bouma Karin Cavallin, SAOB
Examinator:	Richard Johansson
Rapport nummer:	
Nyckelord:	Nysvenska, normaliseringsregler, edit distance, Natural Language Processing, NLP, normalisation, normalisering, ortografi, ortografihistoria, språkteknologi, språkhistoria, ordbok, återskapa, NLTK, regex, SAOB, svenska, Python

Abstract

Svenska Akademins ordbok skall bli färdigställd under 2023. En återstående del är att återskapa äldre ord. Här härleder jag normaliseringsregler för att rekonstruera ortografiska varianter av ord som förekommit från 1500-talet till 1900-talet. I ordboken finns det korta sekvenser kopplade till orden med information om stavning som tidigare förekommit. Problemet att identifiera positionerna i orden där dessa sekvenser skall substitueras in har hittills ansetts olösbart. För detta problem har jag tagit fram en algoritm som bestämmer vilken kombination av normaliseringsregler informationen består av och var den skall substitueras in. Vissa ord är uppdelade i för- och efterled. När något av dessa led saknas, ställs det andra ledet mot huvudordet för att dela detta i två delar. Här används normaliseringsregler och metoden *edit distance*. Normaliseringsreglerna är framtagna från etablerad språkhistorielitteratur. Vid sökning med textord i ordlistan efter tillägg av de genererade orden minskar mätvärdet *precision* något, medan *recall* och *F-score* höjs ordentligt. Sökningen efter nysvenska ord förbättras. En lista med återskapade ord presenteras.

Innehållsförteckning

1 Inledning

Svenska Akademins ordbok skall bli klar under 2023. En del av vad som återstår är att återskapa äldre ord så att de blir sökbara i den digitala versionen av SAOB. SAOB är både en historisk och nutida svensk ordbok som sträcker sig från 1500-talet till nutid. Den har utvecklats i över ett sekels tid.

Genom tiderna har stavningen av ord varierat kraftigt både inom olika tidsperioder och genrer, men även hos samma författare. Man hade egentligen ingen stavningskonvention förrän år 1906 (G. Pettersson, 2005, s. 167).

SAOB återger stavningsvarianter av äldre ord i en väldigt kompakt form med korta sekvenser, som instansierar stavning av ett eller flera fonem. Ett exempel på hur det kan se ut är följande:

TJÄNST: tieniste- (th-, -nn-, -ssth-)

Att identifiera vilka sekvenser i ordet dessa sekvenser skall substitueras med har hittills ansetts olösbart. Det finns ingen gold standard för detta. Det är här mitt arbete kommer in, att hitta regler för att lokalisera vilka sekvenser i ordet som stavningssekvenserna skall bytas med för att återskapa stavningsvarianterna.

Ett sätt att angripa problemet är att se stavningssekvenserna som normaliseringsregler, men eftersom en sådan sekvens ofta realiserar fler än ett fenomen, går det inte att använda normaliseringsregler rakt av, utan man måste först ta reda på vilken kombination av regler som gömmer sig bakom sekvensen. Tag som exempel *-ssth-* ovan, dela upp sekvensen i *ss* och *th*, kombinera två regler *s -> ss* och *t - > th* (bilaga 1). Substituera *ssth* istället för *st* i *tieniste* ovan och vi får: *tienissthe*.

Förhoppningen är att kunna skapa en databas med ord som använder alla stavningsvarianter man hittills hittat, samt alla kombinationer av dem för att generera fler möjliga stavningar.

Flera studier har hittills handlat om att hitta regler för att normalisera äldre texter till nyare texter. Här gör jag istället tvärt om, tar reglerna åt andra hållet och försöker på så sätt återskapa äldre ord från nyare ord. Jag skall inte heller som brukligt normalisera hela ord eller orddelar, utan endast korta sekvenser i ord.

För att lösa problemet kommer jag att använda etablerad forskning i ortografihistoria, kombinatorik och redigeringsavstånd.

1.1 Forskningsfrågor - Problemformulering

- Hur återskapar man äldre svenska ord från nutida svenska?
- Är det tillräckligt med regler uttolkade ur språkhistorien?
- Räcker det med en liten mängd handgjorda normaliseringsregler?
- Går det att rekonstruera ord från SAOB, genom att använda av SAOB givna tidigare stavningsförekomster av bokstäver eller bokstavssekvenser?
- Vilka regler gömmer sig bakom stavningsinformationen?
- Hur mäter man sökningskvaliteten?

1.2 Motivation
Huvudmålet med denna masteruppsats är att möjliggöra sökning av nysvenska ord i Svenska Akademins ordbok. Det skulle underlätta för historiker, lingvister och allmänheten ifall det var möjligt att söka reda på ord ur äldre texter i SAOB. Då SAOB når en stor publik, kan arbetet komma många till godo.

1.3 Bidrag
Mitt ena bidrag är en lista med återskapade nysvenska stavningsvarianter mellan 1500-1900-talet genererade från SAOB:s huvudord och stavningsnycklar (bilaga 4). Denna lista gör det möjligt att söka efter fler ord funna i äldre litteratur digitalt. Bakomliggande algoritmer bygger på kombinatorik, ortografihistoria och redigeringsavstånd.

Mitt andra bidrag är en framtagen samling handgjorda normaliseringsregler, för att ta sig från modern stavning till nysvensk stavning (bilaga 1). Normaliseringsreglerna har mestadels härletts från språkhistoriska påståenden (bilaga 2), men även genom att studera en del äldre varianter av huvudord som redan är klara.

Jag har även tagit fram ett par algoritmer. Dels en algoritm som först hittar vilka normaliseringsregler en given stavningsinformation i SAOB består av med avseende på tillhörande variantord och därefter genererar ord motsvarande alla kombinationer av att sätta in sekvenserna i variantordet, dels en algoritm som delar upp huvudord i förled och efterled, samt kombinerar dessa. Dessa algoritmer är bra att använda för att återskapa olika äldre stavningsvarianter av nutida ord.

2 Bakgrund och tidigare forskning

Hittills har det gjorts flera studier i hur man normaliserar äldre text till nyare text. Detta har ofta gjorts för att kunna använda nutida NLP-verktyg, men också för att länka textord till lexikon. Vissa har använt handgjorda normaliseringsregler, andra har använt tränade. Det är brukligt att använda redigeringsavstånd (eng.: edit distance) för att jämföra hur lika ord är.

Bollmann m.fl. (2011) ville åstadkomma en automatisk mappning från ordformer i tidig nyhögtyska (eng.: Early New High German) till motsvarande moderna ordformer i äldre nyhögtyska (eng.: New High German). För att härleda regler använde de en översättning av Martin Luthers bibel från 1545 och en reviderad översättning av Martin Luthers bibel från 1892. De länkade orden parvis i dessa biblar. Det visade sig att 65 % av orden redan var identiska då bibeln är konservativ till både stavning och terminologi. Resterande icke-identiska ordpar användes för att ta fram normaliseringsregler. De använde härvid Levensteinavstånd samt sorterade reglerna efter uppkomstfrekvens. För att inte generera icke-ord accepterades endast ord som kunde återfinnas i den moderna bibelversionen. Deras metod ökade andelen ord med modern stavning från 65 % till 91%.

Det är svårt att utveckla NLP-verktyg för historisk text eftersom det finns sparsamt av lingvistiskt annoterad historisk data. Ett sätt att komma runt detta är att att först normalisera den historiska texten till mer modern stavning och därefter använda NLP-verktyg för nutida text. Pettersson m.fl. (2012a) visar att det räcker med en liten mängd handgjorda normaliseringsregler för att få stor effekt på resultatet. Reglerna utvecklades från Per Larssons lagsamlingar från 1638 (Edling, 1937).

Pettersson m.fl. (2012b) undersöker hur väl ovanstående normaliseringsregler fungerar på 15 olika svenska texter mellan 1527 och 1812 inom genrerna kyrkdokument och lagsamlingar. De hade en hypotes att normaliseringsreglerna från ovanstående undersökning skulle fungera bättre för lagsamlingar från 1600-talet än för andra genrer och perioder. De antog att emedan det inte fanns några standardiserade stavningskonventioner under denna period borde det vara skillnad på reglernas applicerbarhet mellan texter från olika författare och olika genrer. För detta användes *decision tree*. De märkte dock att de mest frekventa stavningsändringarna var mer eller mindre samma oavsett tidsperiod eller texttyp. De visade att en litet antal handgjorda regler från en enda text var användbara för texter från andra tidsperioder och genrer.

Adesam m.fl. (2021) presenterar en metod för att länka textord till lexikonposter. De använder historiska lexikala resurser för att komma åt tillgänglig kunskap om språkvarianter. De kopplar ett ord i en text till en post i en historisk ordbok och får då både tillgång till den moderna motsvarigheten eller ett lemma som inte längre finns i det moderna språket, och en ordboksdefinition som mer exakt beskriver ordets innehåll. De berättar att hälften av orden i en ordbok inte återfinns i en annan, på grund av att de innehåller olika vokabulär. Ett exempel är sammansättningar som inte längre används, men vars orddelar finns. Då länkas istället huvudordet i sammansättningen från den ena till den andra. En del av uppgiften att koppla textresurser till lexikala resurser är lemmatisering. Lemmatiseringen utförs genom att givet ett tecken gissa vilket lexem det realiserar, baserat på probabilistisk kännedom om hur lemmata motsvarar textformer. De skriver om ett lemma till en textform genom att dela upp lemmat i segment av teckensekvenser och mappar dessa till segment av teckensekvenser

av textordet. Det går att segmentera på olika sätt och även använda tomma sekvenser. Till varje edit-operation tilldelas en sannolikhet. Sannolikheten att observera en viss textform, givet ett lexem, beräknas i termer av sekvenser av flertecken-redigeringar för att komma från lemma till textvariant. Sannolikheten att observera en textform givet ett lexem är då summan av sannolikheter för alla redigeringsoperationer som mappar från lemma till textform.

3 Material

3.1 Material från SAOB

Materialet från SAOB består av en XML-fil med huvudord och tillhörande stavningsparenteser med stavningsvarianter och stavningsinformation. Motsvarande information finns under rubriken *ordformer* i SAOB.

TID: tid (th-, -ii-, -ij-, -j-, -dh),..., tidt (-ii-,-ij-, -dtt, -t)

Varje stavningsvariant har i sin tur en parentes innehållande en mängd korta bokstavssekvenser som representerar olika ortografiska former av ett eller fler ljud. Sekvenserna har ett "-" - tecken före, efter eller på båda sidor om sig. Ifall "-"- tecknet är placerat efter sekvensen skall den substitueras i början av ordet, om det är före, i slutet av ordet och när det är på båda sidor i mitten av ordet. Bokstavssekvensen skall bytas mot motsvarande sekvens i variantordet.

Från exemplet ovan får vi då vi sätter in *th-* i början av ordet *thid, -ii-* i mitten ger *tiid* och *-dh* i slutet blir *tidh*.

Både huvudord och variantord kan ha stavningsinformation knutet till sig
Huvudorden har nusvensk stavning, medan variantorden har nysvensk stavning. Stavningsinformationen gäller i båda fallen hur man kunde stava under nysvensk tid.

Huvudorden står i grundform och är ibland avledda:
En del huvudord innehåller vissa avledningar som *-are* och *-inna* i *TJÄNARE* och *TJÄNARINNA*, medan vissa andra som *TRÄL* inte förekommer avlett till *TRÄLINNA*, däremot till *TRÄLDOM*.

Variantorden kommer från ord som hittats i texter och kan vara både böjda och avledda. Böjda variantord förekommer i bestämd form: *TRIGONOMETRI: -metrie*, i plural: *TVILLING*: *twellingarne* och i bestämd form plural: *TALLRIK: tallerna*

Variantorden är inte bara hela funna textord, utan de är ofta uppdelade i två delar och då inte heller efter morfologi som *tertz-*, *-nell* av *TERTZNELL*.
De hela variantorden är inte avledda, men slutledet av ett delat ord kan bestå av en avledning som det delade variantordet *-erinna* av *TIGRINNA*.

Både huvudorden och variantorden förekommer också som personnamn: *TOMAS: tomas*.

Stavningssekvenserna återger ibland stavningsfenomen på olika sätt *TÖRNE* har variantordet *körne* som har informationen *kiö-*, *TUSENSKÖNA* har variantordet *tusenskön* som har stavningsinformationen *-iö-*. Båda dessa visar att nutida *kö* förr stavades *kiö* (Bilaga 1).

Antalet SAOB:ord är 5177, varav 1769 huvudord, 2089 hela variantord, 753 variantförled, 168 variantmittled (hanteras ej) och 398 variantefterled.
Antalet stavningsinformationer 439 prefix, 943 infix, 558 suffix

3.2 Material för regler

För att ta reda på var i ett ord en substitution skall äga rum behöver man konstruera regler. Dessa regler härleder jag ur påståenden i etablerad språkhistorielitteratur, närmare bestämt ur *Kortfattad svensk språkhistoria* av Gösta Bergman, *Svenska språket under sjuhundra år, En historia om svenskan och dess utforskande* av Gertrud Pettersson och *Svensk ortografihistoria, Från 1200-tal till 1700-tal* av Teleman. Reglerna finns uppställda i en tabell över olika tidsperioder och förekomster i Bilaga 1. Påståendena med uttolkade regler har jag sammanfattat i Bilaga 2. I tabellen i bilaga 1 har jag också listat regler som jag funnit genom egna observationer av stavningsförhållandet mellan SAOB:s huvudord och variantord.

3.3 Material för mätning

För att mäta hur väl algoritmer och regler fungerar behövs ett facit. Facit är framtaget ur de äldre texterna: *Fraktur* (Språkbanken, u.å.-c), *Argus* (Språkbanken, u.å.-a), *Kyrko* (Språkbanken, u.å.-b), *KarlXII* (Språkbanken, u.å.-b), *GVB* (Språkbanken, u.å.-b), *Giftermålsbalken* (Fornsvenska textbanken, u.å.), *Svenska tidningar* 1818-1870 (Språkbanken, u.å.-d), *Svenska tidningar 1871-1906* (Språkbanken, u.å.-e) och *Almqvist* (Litteraturbanken, u.å.).

4 Metod

4.1 Metod för algoritmer

Generella verktyg som jag använder är redigeringsavstånd, bästa väg, sortering, kombinatorik, parsning, regex (Bird m.fl., 2009), NLTK (Bird m.fl., 2009) och matchning.

4.1.1 En algoritm som genererar alla versioner av stavningsinformationer för ett givet ord:

Jag använder handgjorda normaliseringsregler för att återskapa äldre ord från nutida ord (Bilaga 1 och 2).
En stavningsparentes kan innehålla flera stavningsnycklar. För varje nyckel så kan det finnas flera positioner i ordet som uppfyller villkoret att kunna använda nycklarna. Dessa positioner används och kombineras till giltiga lösningar. För varje lösning substitueras sekvensen i originalordet enligt reglerna.

4.1.2 En algoritm som hittar var ett ord delas i förled och efterled:

För att hitta avgränsning för- och efterled utgår jag från algoritmen Levenstein-avstånd men ändrar till en egen form. Levenstein använder i sin grundform kostnaden 1 för redigeringsavstånd och fortsätter hela vägen till slutet (Jurafsky m.fl., 2009). Den väljer billigaste vägen. Min modifierade version använder också kostnader men med olika vikter för olika typer av redigeringsavstånd (eng.: edit distance) och ett tröskelvärde som bryter vidare sökningar. Algoritmen väljer den längsta av kvarvarande vägar som ej brutits. Jag testade mig fram för att hitta lämpliga vikter och tröskelvärden. För att skapa ytterligare ortografiska varianter kombinerar jag återskapade för- och efterled. I de fall ett huvudord endast har förledsvarianter eller enbart efterledsvarianter så används ovanstående tröskelbaserade algoritm för att dela huvudordet i ett för- och efterled. Därefter substitueras alla variantförled eller variantefterled i originalordet.

4.2 Metod för mätning:

För att mäta hur väl algoritmerna rekonstruerar äldre ord genom att generera ord med hjälp av SAOB:s variantord och mina manuellt framtagna normaliseringsregler, tar jag fram ett facit att jämföra med. Med facit menas hur de äldre orden sett ut i verkligheten, dvs i tidigare texter. Jag genererar också en lista med rekonstruerade SAOB:ord. Denna lista skall motsvara en ordbok. Facitorden används för att simulera sökning i de genererade orden Dessa textord har ofta ändelser som inte förekommer i huvudorden, men väl i variantorden. Tag så ett godtyckligt ord ur facit och ett ur listan av genererade ord. Nu söker man efter gemensam matchning ur listan av genererade ord. Vid första tecknet som skiljer så bryts ordet upp från den genererade listan och ordet från facit, i en rot och en ändelse. Om båda ändelserna ingår i var sin lista av godkända ändelser så anses det vara en träff.
Jag delar in sökresultaten i fyra kategorier i en *confusion matrix*: facitordet matchar, facitordet matchar inte, facitordet finns i ordboken och facitordet finns inte i ordboken:

#	facitordet matchar	facitordet matchar inte
facitordet finns		
facitordet finns inte		

För att konstruera ett facit behövs runt 1000 substantiv från olika genrer och tidsperioder. Jag väljer genrerna *biblar, tidskrifter*, *lagar* samt *en allmän samling äldre frakturtryck* från tidsperioden 1541-1917. Texterna hämtas från: *Fraktur* (Språkbanken, u.å.-c), *Argus* (Språkbanken, u.å.-a), *Kyrko* (Språkbanken, u.å.-b), *KarlXII* (Språkbanken, u.å.-b), *GVB* (Språkbanken, u.å.-b), *Giftermålsbalken* (Fornsvenska textbanken, u.å.), *Svenska tidningar* 1818-1870 (Språkbanken, u.å.-d), *Svenska tidningar 1871-1906* (Språkbanken, u.å.-e) och *Almqvist* (Litteraturbanken, u.å.). Därefter listar jag upp *t* - orden automatiskt från dessa texter och sorterar ut substantiven för hand. Jag raderar ord som slutar på vanliga verb- och adjektivändelser, medan jag iakttar viss försiktighet för att inte radera möjliga substantiv. Även ord innehållande "=", "_", "@" tas bort. Jag behåller namn eftersom det förekommer i SAOB. Antalet facitord som kvarstår är 1444.

4.3 Metod för att ta fram datamängd

För att komma åt data i SAOB:s XML-fil läser jag in den med hjälp av *ElementTree* i Python. Sedan letar jag efter relevanta noder. Datan jag får från dessa noder går vidare och analyseras för att hitta huvudord, variantord, och stavningsinformation. De ord som behålls är substantiv, men det finns andra ordklasser. Parsningen av datat är specialiserad på just detta XML-format. Det behövs en parse-kontext för att hålla reda på parenteser. Dessa kommer från olika rader i XML-noderna. Antalet vänster- och högerparenteser går inte i samtliga fall ihop för vissa ord. Detta leder till att jag tar fram en "förlåtande" parser där jag tillåter att vissa parenteser inte alltid går ihop. Det finns även tidsangivelser med årtals-spann som kan innehålla '-' tecken. Eftersom '-' tecknet även används för att dela upp stavningsinformation i förled, mellanled och efterled, så behöver jag ta hänsyn till detta. *word_tokenize* som är en funktion från Pythonbiblioteket *NLTK* (Bird m.fl., 2009) används för att dela upp en rad i dess beståndsdelar. Viss information om stavning som finns i inre parenteser kommer jag dock inte åt i nuläget.

5 Experiment

5.1 Normaliseringsregler

För att tillföra normaliseringsregler för att rekonstruera nysvenska versioner av nusvensk stavning för hand bör man veta en hel del om språkhistoria och då speciellt om ortografi. Jag har studerat ortografi i och sammanfatta valda historiska ortografiska fenomen ur de tre böckerna *Kortfattad svensk språkhistoria* av Gösta Bergman, *Svenska språket under sjuhundra år, En historia om svenskan och dess utforskande* av Gertrud Pettersson och *Svensk ortografihistoria, Från 1200-tal till 1700-tal* av Ulf Teleman,
vilka ligger till grund för normaliseringsreglerna (Bilaga 1 och 2) .

Jag sammanfattar vad tre författarna skrivit om ortografi i hela 187 påståenden (bilaga 2) av samma författare om 14 relevanta fenomen som de har hittat i texter i kronologisk ordning från de fyra tidsperioderna klassisk fornsvenska, yngre fornsvenska, äldre nysvenska och yngre nysvenska mellan årtalen 1225-1906. Information om vilka texter som författarna har tolkat sina påståenden ur och i förekommande fall vem som författat dem finns också sparad.

Från ovanstående 187 påståenden har jag härlett normaliseringsregler. För att det skall gå snabbt och lätt att hitta reglerna har jag också sammanställt en tabell med bara regler under samma tidsperiod och i kronologisk ordning (Bilaga 1). Dessa regler är 88 till antalet.

Det finns en mängd av regler härledda ur påståenden i språkhistorien av typen: *Förr kunde vokaler dubbleras när de var långa*. Om ***v*** lång så ***vv,*** förenklas till: **v** -> **vv** (Bilaga 1).

> *tal (th-, -aa-, -hl)*, *a* i tal byts här mot *aa* och variantordet *taal* genereras

De två senare perioderna är viktiga för att SAOB:s ord påvisar stavningar från dessa perioder. De två tidigare perioderna behövs också tas hänsyn till dels för att det kan finnas stavningar kvar från yngre fornsvenska, dels för att ovanstående författare inte alltid anger tidigaste förekomsten av fenomen.

Ett exempel är inskottskonsonanten *b* mellan *m* och *l* som fanns 1225-1375 (G. Pettersson, 2005, s.98) eller egentligen redan under runsvensk tid (Bergman, 1984, s.25) och fanns i Gustav Wasas bibel. *ml -> mbl* (Bergman, 1984, s.106), men inte behandlas i intervallet 1375-1526. Ifall inte tidsperioden 1225-1375 togs med, skulle man kunna få en felaktig uppfattning om när regeln fanns.

I tabellen förekommer också regler för sje-ljud. Detta ljud hann jag inte studera i språkhistorielitteraturen, utan gjorde en egen tolkning, genom att jämföra SAOB:s huvudord och variantord. och se vilka stavningar man använt förr.

Ett antal regler till härledde jag genom att studera huvudord och variantord i SAOB:s lista. Dessa summerar till 50 och är också inlagda i tabellen (Bilaga 1). Under dessa regler finns variantord som SAOB listat upp, som man kan säga “använder” regeln (Bilaga 1, högra kolumnen).

Jag har även lagt till identitetsregeln, x -> x, så att varje bokstav kan substitueras med sig själv.

Dessa regler kombineras sedan för att matcha sekvenserna i SAOB:s stavningsparenteser.

En del regler förekommer i två varianter för att täcka upp att samma stavning återgetts på olika sätt i SAOB, till exempel kö -> kiö och ö -> iö, se kap. 3.1.

Jag har nu 139 regler inlagda i tabellen ifall man räknar reglerna i kondeserad form. Både till vänster och höger om pilen är det i tabellen (Bilaga 1) en eller fler bokstavssekvenser som motsvarar varandra. Ifall man istället skriver en pil för varje enskilt bokstavssekvenspar blir det 239 regler. I algoritmerna används det senare, en förekomst på vardera sidan om pilen.

5.2 Ordboksord och stavningsparenteser kombinerat med regler:
Med hjälp ovanstående regler som jag tagit fram ur ortografihistorien (Bilaga 1 och 2) tillsammans med huvudord, stavningsparenteser med stavningsvarianter och stavningsinformation från SAOB skall vi återskapa ord som tidigare påträffats i texter (134 av de återskapade orden som sökts upp med textord finns listade med tillhörande ordboksord i bilaga 4).

Ordboksordet *TÅG* har variantorden *tåg* och *tug* med tillhörande bokstavssekvenser i efterföljande parentes.
1: *TÅG: tåg (-åå-, -gh), tug (th-, -uu-, -gh)*
Variantordet *tåg*, sekvensen *-åå-* och regeln ***v*** -> ***vv*** (*å* -> *åå*) (Bilaga 1) ger det nya genererade variantordet *tååg*
tug, th- och regeln *t* -> *th* (Bilaga 1) ger *thug*
tug, *-uu-*, **v** -> **vv** (Bilaga 1) ger tuug
Alla kombinationer av bokstavssekvenserna: *tåg: tååg, tåågh, tågh, tug: thug, tuug, tugh, thug, thuug, thuugh, thugh,*
2: *TÅG*: *tug* (*-w-*, *-gh*)
tug och *-w-*, *u* -> *w* (tabell) ger *twg* (från påståendet: *Långt u stavades med w.*) (Teleman, 2019, s.42) (Bilaga 2)
tug: twg, *twgh*, *tugh*

Huvudorden har ofta variantord som redan “använt” en regel:
TIONDE: tiionda (-ij-)
Huvudordet *tionde* har variantordet *tiionda* som i sin tur har stavningsinformationen *ij*. Här räcker det inte att veta att */i:/* kan stavas med *ii* eller *ij*, som hade varit fallet om man bara skulle rekonstruera äldre ord från huvudord, utan man måste också veta att ii kan stavas ij (Bilaga 1). Med ovanstående regler får vi *tijonda (-ij-)*.

Det finns också stavningsinformationer som representerar två fonem, två stavningsnycklar, och är lite längre, som *-ghl-* och *-ckl-* nedan.
TÄNNLIKA: tenglika (-nng-, -ckl-, -ghl-, -ch-, -ck-)

Man kan tänka sig att reglerna man använder för att substituera dessa stavningsnycklar går från ett element i en startmängd med bokstäver och sekvenser av bokstäver till ett element i en målmängd med bokstäver och bokstavssekvenser.

En del bokstavssekvenser består av flera stavningsnycklar. Vissa av dessa bokstavssekvenser är av typen en sekvens ur målmängden följt av enskild bokstav, t.ex. sekvensen *-ghl-* ovan som skall substitueras i variantordet *tenglika* från huvudordet *TÄNNLIKA*. Normaliseringsreglerna: (*g -> gh*, *l -> l*, *ll -> l*) hämtas från regeltabellen (Bilaga1). *-ghl-* består av elementen *gh* och *l* i målmängden. Här väljs den sekvens i målmängden som passar till motsvarande sekvens i startmängden. Detta blir *gh* som kommer från *g* (*g -> gh*) (Bilaga 1), kvar blir elementet *l* som har två regler till sig (*l -> l*, *ll -> l*). Vi får *gl* och *gll*, men bara sekvensen *gl* finns i *tenglika*, så vi byter ut *ghl* mot *gl* och får *tenghlika*.

Variantordet *tenglika* från *TÄNNLIKA* genererar de nya variantorden:
teghlicha, teghlika, tenghlicha, tenghlika, tenglicha, tenglika, tennglicha, tennglika

5.3 Algoritmer

Här följer tre algoritmer som använts i detta arbete för att kunna generera stavningsvariatianter genom att tillämpa stavningsinformation. Koden är beskriven i Python.

5.3.1 Algoritm som genererar alla versioner av stavningsinformationer:

Introduktion

För att kunna genomföra arbetet genereras flera ord med hjälp av reglerna i kap 5.2., utifrån ett ord, variantord, eller en variantorddel med tillhörande stavningsinformation.
Reglerna går från ett element i en startmängd med bokstäver och sekvenser av bokstäver till ett element i en målmängd med bokstäver och bokstavssekvenser. Ett exempel som kommer användas nedanför för att illustrera algoritmen för substitution är

TIMME, tiim (-ij-, -ijij-)

TIMME är ursprungsordet, *tiim* ett variantord och reglerna *-ij-* samt *-ijij-* är stavningsinformationen som skall tillämpas på variantordet *tiim*. Algoritmen tar variantordet *tiim*, stavningsinformation *-ij-*, *-ijij-* i en lista, samt tre mängder med regler, som tagits fram ur språkhistorien och är listade i bilaga 1. Dessa mängder beskriver hur man kan genomföra ett antal substitutioner från en bokstav, eller flera bokstäver till en eller flera andra bokstavssekvenser för att ta sig från ett modernt ord till ett äldre.
De mängder som tas in är delade i tre grupper. Den första mängden innehåller regler som finns för prefixsubstitutioner. Ett exempel på detta är “*t-*”→ “*th-*”. Den andra och tredje mängden beskriver infix- och suffixsubstitutioner.
Exempelkod för att generera ord:

```
from substitution import generate_words
from rules import create_rules

def main():
    pre, inf, suf = create_rules()
    words = generate_words("tiim", ["-ij-", "-ijij-"], pre, inf, suf)
    print(words)
```

Med resultatet:

```
['tiim', 'tijim', 'tijm', 'tijijm', 'tiijm', 'tijijm']
```

Unika instanser kan enkelt skapas av dessa, och blir då:

```
['tiim', 'tijim', 'tijm', 'tijijm', 'tiijm']
```

Funktionen `create_rules` returnerar tre instanser av en klass som jag skrivit på egen hand. Klassnamnet heter `RuleOrder` och hanterar och samlar en mängd språkhistoriska regler från Bilaga 1 som jag hittat från språkhistorisk litteratur beskriven i kap 3.2 & Bilaga 2.

Substitutionerna kan även göras på förled, mittled, eller efterled som:

```
    words = generate_words("tvivle-", ["-uiff-"], pre, inf, suf)
    print(words)
```

Med resultatet:

```
['tvivle-', 'tuiffle-']
```

Beskrivning av algoritmen bakom: `generate_words`
För att genomföra substitutionerna som beskrivits ovan så arbetar algoritmen i tre övergripande steg:

1. Hitta kandidatpositioner för substitutionerna
2. Lös ut giltiga substitutioner som är konfliktfria
3. Genomför substitutionerna en åt gången

```
def generate_words(word, substitution_rules, prefix_rules, infix_rules,
suffix_rules):
    positions = find_substitions_positions(word, substitution_rules,
prefix_rules, infix_rules, suffix_rules)
    solved_postions = solve_substitutions_positions(positions)
    resulting_words = substitute(word, solved_positions)
    return resulting_words
```

1. Hitta kandidatpositioner för substitutionerna
Kandidatpositionerna gås igenom en och en, där man för varje regel hittar potentiella positioner för att genomföra substitutioner. Resultatet matchas med substitutionstypen för att sålla bort vissa ogiltiga lösningar för bl.a. prefix- och suffixsubstitutioner. Förenklad kod ser ut på följande vis:

```
def find_substitutions_positions(word, substs, pre, inf, suf):
    r = []
    for s in substs:
        pattern = premove_substitution_qualifiers(s)
        ro = select_rule_set_depending_on_substitution_type(pre, inf, suf, s)
        positions = find_positions_from_substitution(word, ro, pattern)
        for p in positions:
            if not discard_pos(word, p, s):
                r.append((p[0], p[1], pattern, get_subst_type(s)))
    r.sort(key=lambda x:x[0]) # sort the replacements by starting position
    return r
```

Kärnan i lösningen ligger i `find_substitutions_positions` .

```
        positions = find_positions_from_substitution(word, ro, pattern)
```

Resultatet av funktionen `find_substitutions_positions` blir en sorterad lista, med avseende på positionerna där man hittat potentiella substitutioner. Elementen i listan som returneras är designad på följande vis: `(position, length, pattern, type)`

Man kan ha flera lösningar på en och samma position och även överlappning kan ske. Detta problem kommer lösaren från steg 2 i den översiktliga beskrivningen ovan att angripa.

Beskrivning av `find_positions_from_substitution`
Denna metod letar upp alla tänkbara positioner där man kan få substitutioner, givet ett ord eller en text, en lista med språkhistoriska regler, samt stavningsinformation.

För att beskriva denna lösning så abstraherar jag den i ett par steg.

- Att hitta giltiga positioner för att söka efter potentiella träffar
- Att ta fram sökpositioner
- Att ta fram kandidatlösningar
- Ta fram giltiga kandidatlösningar
- Skapa en resulterande unik lista av söksträngar
- Ta fram giltiga positioner från söksträngarna

Förenklad pseudokod:

```
def find_positions_from_substitution(text, ro, substitution):
    rules = ro.create_constrained_rules(substitution, text)
    search_positions = find_valid_search_positions(rules)
    candidates = find_candidate_solutions(rules, search_positions)
    valid_candidates = constrain_candidates(candidates)
    search_strings = create_unique_search_strings(valid_candidates)
    positions = find_valid_positions(search_strings, text)
    return positions
```

Att hitta giltiga positioner där man söker efter potentiella träffar
Denna sektion beskriver ungefär hur man löser vad som behöver göras för koden på raden:

```
    rules = ro.create_constrained_rules(substitution, text)
```

När man har stavningsinformationen och en text där man vill utföra substitutionen på så skapar man en abstrakt lösningsmängd. Man kan likna det vid ett träd som ser ut som en trappa, som representeras av en lista av listor som på förhand är bestämd hur den ser ut. Texten används som en restriktion för att minska lösningsmängden och därmed undvika att generera allt för mycket jobb.

För en given regel, t.ex. -*ijij*-, från vårt exempel som vi kommer följa, så skapas följande rader som beskriver den lösningsmängd man söker för ett delmönster från *ijij*. Man vill titta på alla substrängar med alla tänkbara sätt att täcka en given regel.

	pos 0	pos 1	pos 2	pos 3
4 bokstäver	ijij			
3 bokstäver	iji	jij		
2 bokstäver	ij	ji	ij	
1 bokstav	i	j	i	j

Eftersom *ijij* är 4 tecken långt kommer trädstrukturen att ha 4 rader och som max 4 kolumner. Den första raden, som har en kolumn, beskriver *ijij* som ett fullständigt mönster. Den andra raden, med två kolumner, delar upp *ijij* i två delar med 3 tecken var, *iji* och *jij*, dessa väljs genom att för den första kolumnen välja de tre första tecknen på position 0, den andra kolumnen ger tre tecken från position 1. Detta repeteras hela vägen ner tills det bara är 1 bokstav långt substitutionsmönster.

Nu kommer nästa steg som är att hämta alla giltiga regler som finns från denna trappa. Eftersom implementationen av denna konstruktion är gjord i klassen `RuleOrder` så finns alla tillämpliga regler direkt tillgängliga.

Sättet informationen hämtas är att för varje giltig cell i trappan ta fram en lista av bokstavskombinationer som tar oss till målmängden, vilket är specificerat i cellerna i tabellen ovanför. För vårt konkreta exempel med ordet `text="tiim"`, och regeln `substitution="ijij"` och med givna språkregler så får vi vårt resultat:

	pos 0	pos 1	pos 2	pos 3
4 bokstäver	[ii]			
3 bokstäver	[]	[]		
2 bokstäver	[i, j, ii]	[jj]	[i, j, ii]	
1 bokstav	[i, j, ii]	[g, i, j, jj]	[i, j, ii]	[g, i, j, jj]

Dessa kommer nu att avgränsas till vår kontext som är `text="tiim"`. Det betyder att om ett tecken i listorna inte ingår i texten så tas det alternativet bort helt. Detta leder till vår resulterande trappa:

	pos 0	pos 1	pos 2	pos 3
4 bokstäver	[ii]			
3 bokstäver	[]	[]		
2 bokstäver	[i, ii]	[]	[i, ii]	
1 bokstav	[i, ii]	[i]	[i, ii]	[i]

Att ta fram sökpositioner
Här beskrivs lösningen från funktionen `find_positions_from_substitution` för raden:

```
    search_positions = find_valid_search_positions(rules)
```

Sökpositionerna kommer att vara en lista med positioner som beskriver startpositionen i trappan ovanför som `(row, column)`. T.ex. skulle positionen `(1,1)` beskriva raden "**3 bokstäver**" och kolumnen "**pos1**".

De rader och kolumner som sparas är alla element i tabellen som har icke-tomma listor, bortsett från ett specialfall på sista raden där man endast tittar på det första elementet. Detta beror på att den enda giltiga startpositionen för att hitta en giltig lösning måste starta på position 0. Kortfattat så måste man täcka hela intervallet "*ijij*" i den delmängd som är under och till höger om positionen i trädet. Detta betyder att om man t.ex. startar på position 1, så kommer man endast kunna täcka "*jij*", som inte är komplett och därmed ogilitig. Detta resulterar i vårt exempel i en lista:

```
search_positions = [(0,0), (2,0), (2,2), (3,0)]
```

Visuellt så är det:

	pos 0	**pos 1**	**pos 2**	**pos 3**
4 bokstäver	(0,0)			
3 bokstäver				
2 bokstäver	(2,0)		(2,2)	
1 bokstav	(3,0)			

Att ta fram kandidatlösningar
Här beskrivs lösningen från funktionen `find_positions_from_substitution` för raden:

```
    candidates = find_candidate_solutions(rules, search_positions)
```

För varje sökposition skall vi nu hitta kandidatlösningar till problemet.
Detta görs genom att konstruera en filtervektor som talar om vilken del av regeln som är täckt.
Givet en startposition så behöver man linjärt gå igenom alla underrum för att generera regler för de kandidater som täcker mönstret "*ijij*".
Koordinatfiltren (filtervektorerna) för den ovanstående trappan (`0 = False, 1=True`):

	pos 0	**pos 1**	**pos 2**	**pos 3**
4 bokstäver	(0,0,0,0)			
3 bokstäver	(0,0,0,1)	(1,0,0,0)		

2 bokstäver	(0,0,1,1)	(1,0,0,1)	(1,1,0,0)	
1 bokstav	(0,1,1,1)	(1,0,1,1)	(1,1,0,1)	(1,1,1,0)

Filtervektorerna för sökpositionerna blir till sist:

```
[(0,0): (False, False, False, False), (2,0): (False, False, True, True),
(2,2): (True, True, False, False), (3,0): (False, True, True, True)]
```

Där `False` betyder att positionen redan är tagen och `True` att den är öppen för att matcha mot någon lösning. För varje giltig position under startpositionen så genereras ett likadant filter som sedan kontrolleras mot nuvarande tillstånd på filtervektorn. Man får inte ha en lösning där två områden skapar ett överlapp. Detta betyder att om nuvarande filter har `False` på en position så får inte koordinatfiltret ha False på samma position. När man till sist har en filtervektor där alla värden är `False` så har man hittat en match som läggs till i kandidatlistan, sedan återställs filtervektorn till värdet den hade för startpositionen och sökningen fortsätter från punkten efter där man hittade föregående lösning. En kandidatlösning är en lista där man sparar kolumnen och taggar den med sökfiltret i fråga, t.ex. “ij”, samt dess regler. Kandidatlistan för våra sökpositioner kommer efter detta att se ut på följande sätt:

```
candidates = [
  [(0, "ijij", ["ii"])],
  [(0, "ij", ["i", "ii"]), (2, "ij", ["i", "ii"])],
  [(0, "ij", ["i", "ii"]), (2, "i", ["i", "ii"]), (3, "j", ["i"])],
  [(2, "ij", ["i", "ii"]), (0, "i", ["i", "ii"]), (1, "j", ["i"])],
  [(0, "i", ["i", "ii"]), (1, "j", ["i"]), (2, "i", ["i", "ii"]), (3, "j",
["i"])]
]
```

Ta fram giltiga kandidatlösningar

Här beskrivs lösningen från funktionen `find_positions_from_substitution` för raden:

```
    valid_candidates = constrain_candidates(candidates)
```

Detta steg går igenom kandidatlösningen och sållar bort matcher där man potentiellt kan ha fått med sig en tom lista av regler. Detta betyder att lösningen inte kommer att gå att hitta och att den är ogiltig. Till sist sorteras listan med avseende på kolumn. I vårt fall så har vi ingen position i listan som är tom, men det kan alltså inträffa i andra exempel. För detta steg får vi då:

```
valid_candidates = [
  [(0, "ijij", ["ii"])],
  [(0, "ij", ["i", "ii"]), (2, "ij", ["i", "ii"])],
  [(0, "ij", ["i", "ii"]), (2, "i", ["i", "ii"]), (3, "j", ["i"])],
  [(0, "i", ["i", "ii"]), (1, "j", ["i"]), (2, "ij", ["i", "ii"])],
  [(0, "i", ["i", "ii"]), (1, "j", ["i"]), (2, "i", ["i", "ii"]), (3, "j",
["i"])]
]
```

Skapa en resulterande unik lista av söksträngar
Här beskrivs lösningen från funktionen `find_positions_from_substitution` för raden:

```
    search_strings = create_unique_search_strings(valid_candidates)
```

Den här uppgiften går ut på att se till att skapa unika söksträngar. Detta görs genom att expandera alla lösningar baserat på de listor som ingår i `valid_candidates`. Detta följer multiplikationsprincipen och man får t.ex. för `valid_candidates[2]` söksträngarna: `["iii", "iiii", "iiii", "iiiii"]`

Den slutgiltiga unika listan av söksträngar ordet *tiim* med regeln *ijij* blir till sist:

```
    search_strings = ["ii", "iii", "iiii", "iiiii", "iiiiii"]
```

Vanligtvis är många av dessa ej giltiga för ordet i fråga, i detta exempel ordet *tiim*, men det kommer att lösa sig i det sista steget. I praktiken blir det bara `"ii"` som blir giltig.

Ta fram giltiga positioner från söksträngarna
Här beskrivs lösningen från funktionen `find_positions_from_substitution` för raden:

```
    positions = find_valid_positions(search_strings, text)
```

Detta steg är enkelt och tar fram positioner där söksträngar hittas. En regex funktion används här för detta och koden som till sist genererar den slutliga listan av positioner att returnera följer här:

```
def find_valid_positions(search_strings, text):
    r = []
    for search in search_strings:
        positions = [m.start() for m in re.finditer(search, text)]
        for p in positions:
            r.append((p, len(u)))
    return r
```

Man sparar varje position som hittades tillsammans med längden på strängen man letar efter.

För vårt exempel med *tiim* och regeln *ijij* erhålls till den enda positionen:

```
[(1, 2)]
```

Denna position läggs nu till i listan i funktionen `find_positions_from_substitution` en bit upp och taggas med ersättningsmönstret `"ijij"`. Nuvarande lista med positioner ser för tillfället ut så här:

```
    r=[(1,2,"ijij", SubstType.INFIX)]
```

Till sist återstår det nu att repetera precis samma procedur för substitutionsregeln *ij* eftersom vårt exempel var “*tiim* (*-ij-*, *-ijij-*)”.

För att korta ned detta så presenteras endast slutresultatet som då är den sorterade listan med alla positioner och matcher:

```
    r=[(1,1,"ij", SubstType.INFIX),
       (1,2,"ij", SubstType.INFIX),
       (1,2,"ijij", SubstType.INFIX)
       (2,1,"ij", SubstType.INFIX)]
```

Denna lista returneras och avslutar därmed steg 1 i den övergripande beskrivningen.

2. Lös ut giltiga substitutioner som är konfliktfria

Detta steg beskriver steg 2 i funktionen `generate_words`. Det är alltså raden med anropet till `solve_substs_pos` från:

```
def generate_words(word, substitution_rules, prefix_rules, infix_rules,
suffix_rules):
    positions = find_substitions_positions(word, substitution_rules,
prefix_rules, infix_rules, suffix_rules)
    solved_postions = solve_substitutions_positions(positions)
    resulting_words = substitute(word, solved_positions)
    return resulting_words
```

Denna funktion är enkel att utföra eftersom positionerna är sorterade. Man ser till att varje position med tillhörande substitution inte genererar ett överlapp. Varje position har två tänkbara utfall, antingen genomför man substitutionen eller så gör man det inte. Detta leder till ett maximalt resultat med 2^n lösningar, där n är antalet hittade positioner från steg 1. Koden skapar binära permutationer av alla hittade positioner och kontrollerar för varje fall att man inte får ett överlapp med varje lösning. Funktionen ser ut så här:

```
def solve_substs_pos(pos_len):
    # permute all possible combination of rules
    L = len(pos_len)
    Nperm = 1 << L
    result = []
    result.append([]) # append the zero-substitution rule at start and skip it
in the loop
    for i_perm in range(1, Nperm):
        r = []
        for j in range(L):
            if (1 << j) & i_perm:
                r.append(pos_len[j])
        if not has_overlap(r):
            result.append(r)
    return result
```

Vi inkluderar permutationsfallet att inte göra någon substitution alls. För vårt exempel med *tiim* och substitutionerna *-ij-* och *-ijij-* har vi `L = 4`, `Nperm = 16` och får till sist efter man tagit bort alla överlapp resultaten:

```
    result = [
        [],
        [(1, 1, 'ij', SubstType.INFIX)],
        [(1, 2, 'ij', SubstType.INFIX)],
        [(1, 2, 'ijij', SubstType.INFIX)],
        [(2, 1, 'ij', SubstType.INFIX)],
        [(1, 1, 'ij', SubstType.INFIX), (2, 1, 'ij', SubstType.INFIX)]
        ]
```

Detta returneras och avslutar steg 2 i `generate_words`.

3. Genomför substitutionerna en åt gången

Detta steg beskriver steg 3 i funktionen `generate_words`. Det är raden med anropet till `substitute` från:

```
def generate_words(word, substitution_rules, prefix_rules, infix_rules,
suffix_rules):
    positions = find_substitions_positions(word, substitution_rules,
prefix_rules, infix_rules, suffix_rules)
    solved_postions = solve_substitutions_positions(positions)
    resulting_words = substitute(word, solved_positions)
    return resulting_words
```

Funktionen `substitute` tar ordet och listan av konfliktfria positioner och utför substitutionerna en och en i ordning.

Från vårt exempel med *tiim* har vi:

```
    final = [
        [],
        [(1, 1, 'ij', SubstType.INFIX)],
        [(1, 2, 'ij', SubstType.INFIX)],
        [(1, 2, 'ijij', SubstType.INFIX)],
        [(2, 1, 'ij', SubstType.INFIX)],
        [(1, 1, 'ij', SubstType.INFIX), (2, 1, 'ij', SubstType.INFIX)]
        ]
```

Denna lista skall tolkas så att man t.ex. för `final[3]` i ordet `"tiim"` skall från index 1, ta bort 2 tecken och ersätta dessa med `"ijij"`. Raden längst ner gör två substitutioner för ett fall. Vi får slutligen med vårt exempel orden:

```
    words = ["tiim", "tijim", "tijm", "tijijm", "tiijm", "tijijm"]
```

Detta returneras och alla 3 steg har nu genomförts. För att få unika ord så kan man enkelt köra `words = list(set(words))`.

5.3.2 Algoritm för delning av ord med förled:

Introduktion

Man behöver använda förled för att dela ett ord och sedan substituera in andra förled på denna plats och därigenom skapa nya variantord. Det avgörande valet för lösningen är att man har ett brytvillkor för ett angivet tröskelvärde samt tar hänsyn till språkhistoriska regler och identitetsregeln.

Implementation

Huvudfunktionen är `split_words` som tar in ett ord man vill dela `word`, ett förled `prefix`, reglerna från språkhistorien, `rules` samt brytvillkoret `T`. Funktionen returnerar 4 värden i en tupel. `(word_pre, word_suf, prefix_pre, prefix_suf)`.

Algoritmen använder sig av en lista med vägar den hittills valt från språkhistorien `cur_paths` som uppdateras i funktionen `step`. Listan `finished` representerar de vägar som erhållits och avslutats, då kostnaden överstiger tröskelvärdet `T`. Alla vägar kommer att fortsätta tills kostnaden överstiger tröskelvärdet. Man bryter alltså inte vid första lösningen som erhålls.

Till sist väljs den väg som gått längst i ordet och returnerar resultatet. Värdena i både `cur_paths` och `finished` är en lista av tupler, där varje tupel har tre värden med tolkningen: `(pos_word, pos_prefix, cost)` Det sista elementet i listan kommer att peka ut var man är i `word` respektive `prefix` och totalkostnaden. Koden för `split_words`:

```
def split_word(word, prefix, rules, T):
    cur_paths = [[(0,0,0.0)]] # start at positions 0 with cost 0.0
    finished = []
    while cur_paths:
        path = cur_paths[-1]
        cur_paths.pop()

        ongoing, done = step(path, word, prefix, rules, T)
        cur_paths += ongoing
        finished += done

    minpath = None
    for f in finished:
        if not minpath or minpath[-1][2] < f[-1][2]:
            minpath = f

    i_w, i_f, _ = minpath[-1]
    return word[:i_w], word[i_w:], prefix[:i_f], prefix[i_f:]
```

Funktionen `step` utgår från tillståndet `path`, och stegar delningspunkterna ett enda steg. Den returnerar två arrayer, `ongoing`, och `done`. Listan `ongoing` kan ha flera vägval att ta hänsyn till med anledning av de språkhistoriska reglerna. `done` är vägarna som avslutats under stegningen på grund av att kostnaden överstiger tröskelvärdet. Förenklad kod för `step` nedanför. De centrala delarna jag vill lyfta fram är kostnadsfunktionerna, eller redigeringsavstånden, `cost_for_long_rules()` som anropas då man stegar fram längre än ett tecken (t→th), `cost_for_identity_step()` som anropas när man stegar exakt samma tecken i båda orden, och `cost_for_edit_step()` som anropas när tecknen skiljer sig åt och det saknas en språkhistorisk regel för övergången. Funktionerna kan sättas upp på olika sätt, men i denna presentation har jag valt att funktionerna returnerar en konfiguration med konstanta kostnader för de olika momenten.

Notera att funktionen `def step_without_rule():` nedan är en nästad funktion:

```
def step(path, word, prefix, rules, T):
    i_w, i_p, cost = path[-1]
    if i_w >= len(word) or i_p >= len(prefix) or cost > T:
        return [], [path]
    ongoing = []
    done = []
    # find all occurrences of rules that begins with from pattern, i.e. 't',
'k' ... etc
    rules_restr = find_and_constrain_rules(rules, word[i_w:], prefix[i_p:])
    for fr, to_multi in rules_restr:
        # this rule should always proceed since we have both matches... apply
some cost to multirule and continue with code below
        next_cost = cost + cost_for_long_rules() # cost of applying a rule
with multi-step
        if next_cost > T:
            # only add once here, since we terminate before any rule has been
applied or we add duplicates
            if not done:
                done.append(path[:])
        else:
            for t in to_multi:
                cp = path[:]
                cp.append((i_w + len(fr), i_p + len(t), next_cost))
                ongoing.append(cp)
        return ongoing, done # step is done

    def step_without_rule():
        if fr == to:
            next_cost = cost + cost_for_identity_step()
        else:
```

```
            next_cost = cost + cost_for_edit_step()
        if next_cost > T:
            done.append(path)
            return
        path.append((i_w+len(fr), i_p+len(to), next_cost))
        ongoing.append(path)

    fr = word[i_w]
    to = prefix[i_p]
    if fr in rules:
        actually_stepped, o, d = step_with_rule(fr, to, rules, path,
cost_for_rule())
        ongoing += o
        done += d
        if not actually_stepped:
            step_without_rule()
        return ongoing, done
    else:
        step_without_rule()
        return ongoing, done
```

Exempel, torsdag
Nedanför finns ett exempel på delningen av `"torsdag"` med förledet `"thors-"`.
Innan man anropar `split_words` så skall avslutande ‘-’ tecken tas bort:

```
from rules import create_rules
from subst_old import split_words
def main():
    rules = create_rules()
    print(split_words("torsdag", "thors", rules, T))
```

Utskrift:

```
('tors', 'dag', 'thors', '')
```

Detta betyder att ordet `'torsdag'` med förledet `'thors-'` delades i två delar, `'tors'` och `'dag'`, samt delningsordet i `'thors'` och `''`. Detta kan man använda t.ex. för att konstruera det alternativa ordet `'thorsdag'` genom att välja förledet `'thors'` och till det addera det andra delningsordet `'dag'` från torsdag, som funktionen returnerat.

5.3.3 Algoritm för delning av ord med efterled:

Introduktion
Ibland behöver ett efterled dela ett ord för att sedan substitueras in i ett ord och därmed skapa ett nytt variantord. För att lösa detta problem togs denna metod fram i arbetet. Ett

centralt val som gjordes var att använda delning av ord med förled, `split_words`, i lösningen. Man har därmed ett brytvillkor för ett angivet tröskelvärde samtidigt som man tar hänsyn till språkhistoriska regler.

Beskrivning av metod
För att hitta en gemensam delningspunkt för ändelserna så har man en poäng-vektor. Antalet kolumner för vektorn är lika med längden på ordet.

För att ta fram en gemensam delningspunkt så räknas en rad-poäng ut för varje efterled. Denna adderas sedan till poäng-vektorn och man får en slutgiltig poäng-vektor.

För att hitta delningspunkten så väljs den kolumn som har högst poäng. Sökningen påbörjas bakifrån för att premiera att man klipper längre bak i ordet för de fall då man får samma max poäng på flera positioner.

Implementation
Huvudfunktionen är `find_division` som tar in ett ord man vill dela `word`, en lista med efterled `word_endings`, reglerna från språkhistorien, `rules` samt brytvillkoret `T`. Funktionen returnerar indexet för den position i `word` där delningspunkten hittats.

Tröskelvärdet och vikterna har satts så att `split_words` bryter om man genomför en edit operation:

```
def find_division(word, word_endings, rules, T):
    s = [0] * len(word)
    for e in word_endings:
        se = eval_scores(word, e, rules, T)
        s = [sum(x) for x in zip(s, se)]
    if sum(s) == 0:
        return len(s) # no solution
    imin = len(s)-1
    i = imin
    while i >= 0:
        if s[i] > s[imin]:
            imin = i
        i -= 1
    return imin
```

Funktionen `eval_scores`:

```
def eval_scores(word, ending, rules, T):
    scores = [0] * len(word)
    for i in range(len(word)):
        cur_word = word[i:]
        cur_pre,_, _,_ = split_words(cur_word, ending, rules, T)
        scores[i] = len(cur_pre)
```

```
    return scores
```

Exempel, tålamod
Nedanför finns ett exempel på delningen av ordet *tålamod* med efterled som är tagna från ordlistan SAOB bidragit med.
Innan man anropar `find_division` så skall man ta bort inledande ‘-’ tecken:

```
    word = "tålamod"
    endings = [
        "mo", "mod", "mod",
        "modh", "mood", "moodh",
        "modt", "modt", "mot"
    ]
    pos = find_division(word, endings, rules, T)
    print(f"{word[:pos]}-{word[pos:]}")
```

Utskrift:

```
tåla-mod
```

Poäng-vektorerna följer, en rad för varje delningsord:

```
[0, 0, 0, 0, 2, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
[0, 0, 0, 0, 3, 0, 0]
```

Resulterande poäng-vektor:

```
[0, 0, 0, 0, 26, 0, 0]
```

Exempel, tacksägelse
Nedan finns ett exempel på delningen av ordet tacksägelse med efterled som är tagna från SAOB:

```
    word = "tacksägelse"
    endings = [
        "segelse", "seielse", "seigelsen",
        "seilsens", "seyilse", "siellse",
        "syelse", "sägelse", "säjelse"
    ]
```

```
    pos = find_division(word, endings, rules, T)
    print(f"{word[:pos]}-{word[pos:]}")
```

Utskrift:

```
tack-sägelse
```

Poäng-vektorerna:

```
[0, 0, 0, 0, 7, 0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 1, 0, 5, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 7, 0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 7, 0, 0, 0, 0, 2, 0]
```

Resulterande poäng-vektor:

```
[0, 0, 0, 0, 31, 0, 5, 0, 0, 16, 0]
```

5.4 Facit

För att kunna mäta kvalitet på uppkommet sökresultat behövs ett facit. För att tillverka ett facit behövs runt 1000 substantiv från olika genrer och tidsperioder.
Jag valde genrerna *biblar, tidskrifter*, *lagar* samt *en allmän samling äldre frakturtryck* från tidsperioden 1541-1917. Texterna är hämtade från: *Fraktur* (Språkbanken, u.å.-c), *Argus* (Språkbanken, u.å.-a), *Kyrko* (Språkbanken, u.å.-b), *KarlXII* (Språkbanken, u.å.-b), *GVB* (Språkbanken, u.å.-b), *Giftermålsbalken* (Fornsvenska textbanken, u.å.), *Svenska tidningar* 1818-1870 (Språkbanken, u.å.-d), *Svenska tidningar 1871-1906* (Språkbanken, u.å.-e) och *Almqvist* (Litteraturbanken, u.å.). Därefter listade jag upp *t* - orden automatiskt från dessa texter och sorterade ut substantiven för hand. Jag raderade ord som slutade på vanliga verb- och adjektivändelser, medan jag iakttog viss försiktighet för att inte radera möjliga substantiv. Även ord innehållande “=”, “_”, “@” togs bort. Jag behöll namn eftersom det förekommer i SAOB. Antalet facitord som härefter återstår är 1444.

5.5 Mätmetod

När jag tar ställning till om det är en träff eller inte med min kontextfria sökordslista, har jag en förenklad modell, där jag räknar *träffar* med avseende på ifall ordet skulle kunna ha betytt detta.

Mätningen går till så att man tar ett facitord, som simulerar en sökning, och jämför det med ord från den genererade listan som föreställer en ordbok. För att avgöra om de stämmer överens, d.v.s. att man kunde hitta det sökta ordet i listan, behöver man ta hänsyn till att de genererade orden ofta, men inte endast förekommer i grundform, medan facitorden ofta är böjda eller avledda.

Jag mäter med tre metoder, en med exakt matchning, en med listor med godkända ändelser för sökord och ordboksord, samt en med godkända listor med ändelser och avledningar.

5.5.1 Mätmetod med godkända ändelser och avledningar

Här testar jag en lösning där jag sammanställer en mängd godkända avledningar och ordböjningar (Bilaga 3). Om ett facitord är lika ett genererat ord fram till en viss punkt och resten av facitordet är lika ett ord i den godkända ändelsemängden, anses det hittat. Det genererade ordet kan i sin tur ha en slutsekvens som måste kapas innan man söker i mängden av godkända slutsekvenser.

Vid valet av tillräcklig ändelsemängd har jag utgått från *Kontrollens manual 2022.1*, ett redaktionellt arbetsdokument vid SAOB, Lund, och har därifrån hämtat och sammanställt denna med avseende på deklinationer i bilaga 3. Ändelserna summerar till: avledningarna: *-a, -an, -ande, -are, -arinna, -dom, -else, -eri, -erska, -het, -ing,- nad, -ning, -sel, -skap,* bestämda och obestämda pluralformerna: *-arna, -arne, -arnar, -arner, -arna,-orna, -erna,* pluralformerna med r-bortfall: *-ana, -ane, -anar, -aner, -ona, -ena, -er, -or, -er,* samt formerna för ental i bestämd form: *-n, -en, -et*. Dessutom har jag lagt till alla dessa former i genitiv, med *-s* på slutet. Efteråt har jag kompletterat med *-inna*, de äldre formerna *-erne, -ene, -a, -es,* samt varianterna *-erij, -heet*. Det genererade ordet kan kapas före förekomst av *-a, -e, -er, -s*.

En falsk träff är när man tillåter ändelser som *-ande* för facitorden och *-er* för huvudorden, till exempel *tigande* som kommer av *tiger*.
En annan falsk träff med *-het* är *tomhet*. Detta ord borde inte hittats då *tom* är ett adjektiv, men i SAOB:s lista finns substantivet *tom*.

Det är dock inte relevant i nuläget att mäta med godkända avledningar. Eftersom orden jag genererar nya ord från är substantiv, kan inte avledda ord från andra ordklasser bildas.

5.5.2 Mätmetod med godkända ordböjningar

Här simulerar vi precis som ovan sökning med ett facitord och jämför det med ett ord från den genererade listan. Orden stämmer här överens ifall det som skiljer dem åt är att facitordet har en ändelse i listan med godkända ordböjningar (bilaga 3) och det genererade ordet eventuellt kan kapas före *-a, -e, -er, -s*. Denna metod är relevant eftersom facitorden oftare är böjda än variantorden i ordboken.

5.5.3 Mätmetod med exakt matchning

Mätningen går till som ovan, att man tar ett facitord som simulerar en sökning, och jämför det med ord från den genererade listan som föreställer en ordbok. För att avgöra om de stämmer överens, d.v.s. att man kunde hitta det sökta ordet i listan godkänns här bara exakt matchning. Detta för att variantorden som algoritmerna genererar ord från kan vara både avledda och böjda, liksom facitorden från historiska texter.

6 Resultat

Här följer resultatet av att återskapa nysvenska ord från nusvenska ordbokord med hjälp av stavningsinformation från SAOB och handgjorda normaliseringsregler:

6.1 Antal ordboksord före och efter generering

Detta har vi innan ordgenereringen:

Innan generering är antalet facitord 1444. Egentligen 814 eftersom 630 ej kan hittas med substantiv, se kap. 6.5).

Antalet SAOB:ord är 5177, varav 1769 huvudord, 2089 hela variantord, 753 variantförled, 168 variantmittled (hanteras ej) och 398 variantefterled.

Antalet stavningsinformationer är indelade i: 439 prefix, 943 infix, 558 suffix.

Detta genereras:

Antal genererade ord är 6168, varav genererade ord med enbart stavningsinformation är 3587, genererade med delning med hjälp av förled är 394, genererade med delning med hjälp av efterled är 186, genererade permutationer av för- och efterled 2001.

Man ser att det genererade antalet ord ovan är ett rimligt antal. Alternativet att villkorslöst tillämpa samtliga normaliseringsregler på alla ordboksord skulle resultera i ett exponentiellt antal genererade ord. Om varje bokstav i ett ord har en normaliseringsregel, så genereras 2^(antal bokstäver i ordet) nya ord. Antal ord är 5177, genomsnittlig ordlängd är 6. Då hade det genererats 2^6 * 5177 = 331328 ord.

6.2 Lista med återskapade ord som återfunnits i texter

Vid sökning i listan av ord som återskapats med hjälp av normaliseringsregler och stavningsinformation kan 104 (kap. 6.5) av dessa genererade ord hittas med exakt matchning (kap 5.5.3). Ytterligare 30 ord (en ordböjning per ord) som hittats med metoden som tillåter ordböjning (kap. 5.5.2) är tillagda i slutet på listan. Dessa återskapade ord har kopplats till respektive ordboksord och finns listade i bilaga 4.

De 30 återskapade ord som hittats genom att tillåta ordböjning har inte kopplats med lika hög säkerhet som orden som funnits med exakt matchning, eftersom tillägget att tillåta ändelser kan introducera fel.

6.3 Framtagna normaliseringsregler:

Jag har tagit fram 88 handgjorda normaliseringsregler från påståenden som är uttolkade ur språkhistorielitteratur. Dessa är beskrivna i kapitel 5.1 och redovisade i bilagorna 1 och 2.

6.4 Konstruerade algoritmer:

Tre algoritmer konstruerades för ordegenereringen. En algoritm (kap. 5.3.1) letar upp vilka normaliseringsregler stavningsinformationen från SAOB innehåller, samt var i ett ord informationen skall placeras. Denna algoritm kombinerar också alla kombinationer av stavningsinformation för att skapa fler möjliga varianter. Två algoritmer delar upp orden med SAOB:s förleds- och efterled-varianter för att generera ytterligare ord (kap. 5.3.2 och kap.5.3.3).

6.5 Kvalitetsmätning:

Här är resultatet av att mäta hur väl det går att söka i en lista med genererade ord med ord från facit som simulerar sökningar med textord (kap. 5.5.).

Att ett sökt ord “hittas” i en genererad lista, kan vara en av två anledningar, antingen att det har ett släktskap med ett genererat ord eller att det är en falsk träff.
Att ett ord inte hittas kan också bero på två orsaker, antingen att mina algoritmer och regler inte fungerar eller är tillräckliga, eller att ordet inte finns i ordboken och därmed inte kunnat generera en äldre version.

Listan med sökord är kontextfri, så jag har en förenklad metod, där träffar räknas med avseende på om ordet skulle kunna betyda detta.

Jag mäter med tre olika metoder. En metod använder exakt matchning. Två metoder har listor med godkända ändelser för sökord och ordboksord. Orden jämförs här fram till den punkt de skiljer sig åt och godkänns ifall den överskjutande delen finns i respektive lista av godkända ändelser. Den ena av dessa metoder tillåter ordböjning och den andra ordböjning och avledning.
En match räknas som minst en träff.

Metoderna i tabellform för jämförelse:
Nedan har jag sammanställt resultaten i två tabeller som anger antalet hittade och ej hittade facitord, före respektive efter ordgenerering, med avseende på vilken metod som använts.

Antal sökord som hittas i SAOB med avseende på metod innan tillägg av genererade ord

#	Hittade, utan genererade ord	Ej hittade utan genererade ord	Andel hittade i %
Ändelser och avledningar (kap. 5.5.1)	652	792	45,2
Ändelser (kap. 5.5.2)	608	836	42,1
Exakt matchning (kap. 5.5.3)	259	1185	17,9

Antal sökord som hittas i SAOB, med avseende på metod, efter tillägg av genererade ord

#	Hittade, med genererade ord	Ej hittade, med genererade ord	Andel hittade i %
Ändelser och avledningar (kap. 5.5.1)	821	623	56,9
Ändelser (kap. 5.5.2)	774	670	53,6
Exakt matchning (kap. 5.5.3)	363	1081	25,1

Metoden med ändelser är mest relevant, emedan metoden med avledningar är intressant först när andra ordklasser än substantiv behandlas och då metoden exakt matchning inte hittar ordböjningar som är vanliga i textord, därför skall jag börja med att titta närmare på dess resultat nedan, de andra gås därefter endast översiktligt igenom.
Tillåt ordböjning:
Vid sökning där vi tillåter ordböjning (kap. 5.5.2) redovisar vi resultaten i en *confusion matrix*. För varje facitord undersöker vi två saker: matchar det i ordbokslistan med den aktuella

matcningsmetoden (ja eller nej), och finns facitordet faktiskt i ordboken (ja eller nej). Vi har då följande kombinationer:

1. Facitordet matchar och det finns faktiskt
2. Facitordet matchar men det finns faktiskt inte
3. Facitordet matchar inte, fast det faktiskt finns
4. Facitordet matcar inte, och det finns faktiskt inte heller

Resultatet av sökning före och efter generering med tillåtelse av ordböjning ges i tabellerna nedan. Antalet ord i ordlistan ökar efter återskapande av nysvenska substantiv.

Ordböjning, SAOB:s huvudord och variantord innan ordgenerering

#	Facitordet matchar	Facitordet matchar inte	
Facitordet finns	591	187	778
Facitordet finns inte	17	649	666
Totalt 1444	608	836	

Ordböjning, SAOB: huvudord, variantord och genererade ord

#	Facitordet matchar	Facitordet matchar inte	
Facitordet finns	738	40	778
Facitordet finns inte	36	630	666
Totalt 1444	774	670	

Skillnaden mellan antalet ord som finns och matchar före och efter generering är 173 (754 - 581), vilket innebär att 173 av de genererade orden är funna. Härav drar jag också slutsatsen att det har fungerat bra att använda handgjorda normaliseringsregler för att komma åt stavningsnformation inför generering. Att *precision* nedan också ökar, om än lite grann, medan *recall* och *F-score* fortfarande är höga, efter ordgenerering, motsäger inte detta.

Precision, andelen matchande facitord som faktiskt finns i ordboken, TP / (TP+FP):
Före generering: 591 / (591+17) = 0.9720
Efter generering: 738 /(738+36) = 0.9535
Här kan man se att andelen facitord som finns i ordboken ökar efter generering. Att skillnaden inte blir större beror på att en del facitord som tillhör andra ordklasser än substantiv felaktigt matchar genererade substantiv beroende på metoden *tillåt ordböjning*. Dessa hamnar i rutan matchar, men finns inte i ordlistan bland substantiv. T.ex. *twaget*, matchar mot *twaga* från *tvaga*.

Recall, andelen i ordboken förekommande facitord som också matchar, TP / (TP +FN):
Före generering: 591 / (591 + 187) = 0.7596

Efter generering: 738 / (738 + 40) = 0.9486
Anledningen att *recall* går ned är att det introduceras en del programvarufel vid generering, så att de ord som borde byggts ihop, inte gör det se nedan. Vissa genererade ord hittas inte för att de har en ändelse som inte finns i ändelselistan över godkända ändelser (bilaga 3).

F-Score: 2 x (Precision x Recall) / (Precision + Recall):
Före generering: 2 x (0.9720 x 0.7596) / (0.9720 + 0.7596) = 0.8528
Efter generering: 2 x (0.9535 x 0.9486) / (0.9535 + 0.9486) = 0.9510

Distribution av antalet träffar:
Det kan vara intressant att titta på distributionen av antalet träffar, eftersom tabellerna ovan endast anger en match då det är minst en träff, medan distributionen av träffar tittar på fördelningen av alla träffar. Nedan ser man att det finns 367 facitord som fått 1 träff, 135 facitord har 2 träffar, o.s.v.

Distribution av antalet träffar, vid tillåtelse av ordböjningar:

Antal träffar	1	2	3	4	5	6	7	8	9	10	11	13	15	18
Antal gånger	367	135	67	80	58	26	18	13	4	2	1	1	1	1

Antalet träffar i medeltal blir 2,52.

Antalet träffar blir fler än antalet hittade facitord, eftersom ett ord både kan förekomma i SAOB som huvudord eller variantord, eller vara genererat av SAOB:s huvudord eller variantordord.
Som exempel får sökordet *troon* får tre träffar:
troon finns som variantord SAOB:s huvudord *TRAN*.
troo genererat av SAOB:s huvudord TRO,
troo genererat av variantordet *tro* från huvudordet *TROD*.

630 Kan ej hittas av nedanstående anledningar (sant kan ej hittas):
En del ord kan inte hittas för att de inte finns med i ordlistan eller saknar stavningsinformation. Från XML-filen jag utgick ifrån skulle den delmängd som motsvarar rubriken *ordformer* i SAOB bearbetas. Efterleden till sammansättningar finns inte under rubriken *ordformer*, därför behandlas inte sammansättningar (315 stycken). Ifall jag hade hanterat andra ordklasser än substantiv, hade de 119 avledningningarna nedan hittats. Det förekommer namn i ordboken, men de flesta är inte upptagna.
315 sammansättningar (thordönsstrålar, tidehwarfwens, tiensteandar, tänckeskrifter,...)
71 namn (*thimotheus, therese, titus*)
18 latin (*textus)*
2 franska (*tout*)
3 engelska (*trait*)
15 växter (*tragopogon*)
8 djur (*tahltrasten*)
12 special (*tilb, timej, tomtg,* …)
119 adjektiv, verb eller räkneord med avledning som bildar substantiv
14 substantiverade adjektiv

18 andra ordklasser än substantiv
13 ordform saknas i ordbok
12 stavningsinformation saknas i ordbok beroende på förledet *till*
9 stavningsinformation saknas i ordbok, (4 olika stavningsinformationer)
1 ord saknas i ordbok (*trident*)
40 Kan ej hittas p.g.a. följande fel (falskt kan ej hittas):
En del fel borde hittats, men gör inte det beroende på mätmetod eller programvarufel:
13 grammatikfel (saknas ändelse i egentillverkad ändelselista)
7 programvarufel (hittar ej variantord)
18 programvarufel (7 olika parentesfel)
2 normaliseringsregler saknas (1 regel)

Tillåt ordböjning och avledningar:
När man tillåter ordböjningar och avledningar vid sökning med facitorden i SAOB:s huvudord och variantord innan generering, blir antalet hittade facitord 683 och antalet ej hittade 770.

Vid tillägg av de genererade orden till SAOB:s huvudord och variantord blir antalet hittade facitord 852 och antalet ej hittade 601, varav 40 borde hittats.

Distribution av antalet träffar:
Antalet träffar blir blir fler än antalet sökord eftersom ett sökord kan matcha variantord som är genererade från flera olika huvudord. Här anges hur många gånger man får ett visst antal träffar.

Distribution av antalet träffar, vid tillåtelse av avledningar och ordböjningar:

Antal träffar	1	2	3	4	5	6	7	8	9	10	11	13	15	18
Antal gånger	362	156	81	90	61	26	20	14	5	2	1	1	1	1

Antalet träffar i medeltal blir 2.57.

Denna metod visar sig dock vid närmare eftertanke inte vara bruklig i det här arbetet emedan jag bara arbetar med substantiv. Det är ingen idé att ta hänsyn till avledda ord, då dessa förvisso är substantiv, men avledda från andra ordklasser. Skulle man t.ex. ta med verb, kunde metoden vara intressant.

Exakt matchning:
Vid sökning med facitorden med metoden exakt matchning (kap. 5.5.3) får vi följande resultat:

Antal hittade ord i enbart SAOB:s huvudord och variantord är här 259 och antal ej hittade ord är 1185.

Antal hittade ord med tillägg av genererade ord till SAOB:s huvudord och variantord är 363 och antal ej hittade ord är 1190.

Här är skillnaden mellan antal hittade ord före och efter generering 104, vilket innebär att 104 av de genererade orden matchas med exakt matchning.

Distribution av antalet träffar, vid exakt matchning:

Antal träffar	1	2	3	4	5
Antal gånger	260	67	21	10	5

Antalet träffar i medeltal blir 1.44.

Det blir ett väldigt stort antal ord som ej hittas. Metoden är egentligen inte så intressant eftersom de flesta sökorden är böjda eller avledda, samtidigt som de flesta ordboksorden står i grundform. Metoden exakt matchning är mest med som referens till de andra två metoderna.

7 Diskussion

7.1 Beredning av materialet

Det tog lång tid att bearbeta datan i XML-filen så att det gick att plocka ut relevant information för orden. Det finns säkert bättre metoder än att leta efter parenteser som hänger ihop. Viss information om stavning fanns i inre parenteser i stavningsinformationen. Denna information lyckades jag inte komma åt i nuläget. Det får bli ett framtida arbete

7.2 Stavningsparentes

Man kunde tänka sig att istället för att använda de av SAOB angivna representationerna av ortografiska varianter av olika bostavssekvenser helt enkelt kombinera alla möjliga kombinationer av resultaten av att använda reglerna direkt utan att se efter om det finns en stavningsinformation. Söka igenom reglerna för att se vilka som går att använda dels bokstav för bokstav, dels för bokstavssekvenser. Tillämpa varje möjlig regel från nutida realisering till äldre stavning av ljud och spara alla kombinationer av äldre stavningar. .
Det skulle då bildas onödigt många “ickeord” (kap. 6.1).

7.3 Algoritmer

Idén att generera alla kombinationer utifrån en mängd stavningsnycklar är intressant. Det är dock inte huruvida algoritmen kan kombinera alla kombinationer av stavningsnycklar som sedan mäts, utan just ifall idén att generera alla kombinationer av stavningsnycklar är hållbar. Jag fick tänka på detta när jag räknar falska träffar. En falsk träff är i så fall ett genererat ord som har en kombination av nya stavelser som gör att ordet byter innebörd, eller kan härledas till ett annat huvudord, än det ord det genererats från.

7.4 Facit

Det är tidskrävande att leta efter texter som digitaliserats, de flesta är endast fotograferade. Därför har jag i första hand valt större färdiga textsamlingar och sedan fyllt på för att täcka fler tidsperioder och genrer.
Det är förvisso betoning på ord från 1700-talet, men det torde inte göra någonting då Pettersson et. al visade att de mest frekventa stavningsändringarna var ungefär samma oavsett tidsperiod och genre (E. Pettersson m.fl., 2012b) (kap 2.).

7.5 Påståenden och regeltabell

Språkhistoria: vissa fenomen i språkhistorien kan antas följa med sedan lång tid tillbaka, därför har jag tagit med flera århundraden före 1500 -talet. Man kan tycka att jag tagit med ortografisk historia från lite väl långt tillbaka, men vissa tidiga fenomen återges först betydligt senare. Någonstans behöver gränsen dras och jag tror inte att SAOB har tänkt att det skall bli möjligt att söka på runor.
Eftersom Pettersson m.fl., (2012b) använde *decision tree* och visade att de mest frekventa stavningsändringarna var samma oavsett tidsperiod, beslutade jag mig för att det räckte med att ta fram handgjorda normaliseringsregler för hela tidintervallet utan avseende på årtal.

Det är säkerligen ett överskott av regler i regeltabellen (Bilaga 1), speciellt bland dem jag lagt dit genom studier av SAOB:s huvudord och variantord. På grund av tidsbrist har jag inte testat vilka som verkligen behövs. Det är endast en regel jag inte hittar i min regeltabell.
Jag hittar inte någon regel för att byta ut *äj* variantefterledet *-säjelse* med informationen *-äy-* från *TACKSÄGELSE*.

7.6 Mätmetod

Jag har valt att lägga alla tillåtna ändelser på huvudord och sökord i två olika mängder istället för att tillfoga dem på slutet av orden i grundform.

Min mätmetod att jämföra ett sökord med ett genererat ord och en godkänd ändelselista är enkel, den genererar inga nya ord, som skulle varit fallet ifall jag tillfogade dem på slutet. Jag hade kunnat kunnat använda *finite-state* kompilatorn Foma (Huldén, 2009) och bygga ihop alla kombinationer av ordböjningar och avledningar vid ordgenereringen, men jag tyckte inte att det passade riktigt här då variantorden som jag genererar ord från ibland redan är böjda eller avledda. Huvuduppgiften var att generera ord med befintlig stavningsinformation, till skillnad från att dessutom generera nya ord med alla tänkbara ändelser.

Mätmetod med godkända ändelser och avledningar.

En begränsning att ha listor med godkända avledningar är att den nu missar substantiv bildade av adjektiv med produktiva ändelser, vilket gör att andelen hittade ord sjunker.

Ett exempel är sökordet *tacksamhet*. Substantivet *tack* finns i den genererade listan av substantiv, men inte adjektivet *tacksam*. Således missas substantivet *tacksamhet* med den produktiva ändelsen *-het*.

Denna begränsning har dock uppstått eftersom jag hållit mig till substantiv och skulle inte innebära ett problem ifall jag inkluderade alla ordklasser.

Mätmetod med godkända ordböjningar

Flera av facitorden är böjda, men ordboksorden är ofta oböjda, därför kan den passa med en metod som tar hänsyn till ordböjning.

Mätmetod med exakt matchning

Huvudorden är inte böjda medan variantorden som de genererade huvudorden utgår ifrån och ord från historiska texter är böjda och avledda. Om det finns ett historiskt ord som har samma stavning som nutida ord och variantordet som det genererade ordet utgår ifrån inte är böjt, skulle det historiska ordet inte hittas. Det skulle ge ett falskt resultat. Då flera facitord är böjda missar denna metod för många ord för att vara relevant.

7.7 Framtida arbete

Ett framtida arbete som är önskvärt från SAOB:s sida är att undersöka om det går bra att använda samma regler för ord som börjar på andra bokstäver än *t*. Min förmodan är att det borde gå lika bra då inte något i vare sig algoritmer eller regler bygger på att orden börjar på *t*, men det återstår att se. Det borde också gå bra att använda reglerna för andra ordklasser. Ett annat är att testa på andra ordklasser än substantiv. Då skulle man även kunna hitta substantiv som bildats av avledningar.

Det kan vara intressant vid fortsatta studier att kunna röra sig fram och tillbaka i tabellen med normaliseringsregler (Bilaga 1), och testa att använda *decision tree*, ett beslutsträd, med avseende på årtal och källa, när rätt regel skall väljas. Tabellen i bilaga 1 är förberedd för detta.

En inläsning jag inte hann arbeta med var stavningsinformation i form av extra parentes inuti parentesen: *tjyv (ti-, -ÿ-, -f(f))*. Detta skulle kunna läsas in och behandlas också.

Att använda maskininlärning skulle kunna vara en möjlig förbättring av normaliseringsregler, så att fler äldre ordformer genereras, vilket skulle underlätta sökning på äldre ord.

För att hitta orden i sammansättningar skulle man kunna använda *delningsalgoritmen* beskriven i avsnitt 5.3.2. Den delar av ordet i ett förled och ett efterled. Man kunde sedan länka äldre sammansättningar till huvudordet precis som Adesam et. al. gör. När jag skall

söka upp huvudorden i sammansättningar hade det varit bra med en lista av “sammansättningsordslut”, egentligen en lista av ord som börjar på andra bokstäver än *t* att jämföra ordsluten i sammansättningar med, för att se om *delningsalgoritmen* delar så att ordslutet också är ett helt ord. Här har dock brist på tid varit en begränsning, så att jag inte hunnit så långt.

8 Slutsats

Här svarar jag på inledande forskningfrågor (inledningen) i turordning:

Hur återskapar man äldre svenska ord från nutida svenska?

Det går bra att använda handgjorda normaliseringsregler för att gå från nutida till nysvensk stavning. Det är regler som jag tagit fram ur erkänd språkhistorielitteratur (kap. 3.2) och sammanställt i bilaga 1 och 2. Reglerna fås genom att titta på ortografiska skillnader. Som exempel kan nämnas, dubbel- istället för enkelskrivna vokaler, beteckning av långt i med ij, stavning av långt u med w och v med ffu. Detta kan skrivas om som reglerna a -> aa, i -> ij, u-> w och v -> ffu. Tag så ett nusvenskt ord. Titta på en sekvens av bokstäver i ordet som motsvarar ett fonem, se om det finns en normaliseringsregel för sekvensen, substituera in den nysvenska stavningen av samma fonem givet regeln i ordet (kap 5.2).

Är det tillräckligt med regler uttolkade ur språkhistorien?

Det är nästan tillräckligt med regler uttolkade ur språkhistorien. Jag lade bara till en andel regler som jag fått genom att manuellt studera skillnaden mellan SAOB:s huvudord och stavningsvarianter (Bilaga 1).

Räcker det med en liten mängd handgjorda normaliseringsregler?

Ja, det räcker med ett par hundra handgjorda nomaliseringsregler (Bilaga 1 och 2).

Går det att rekonstruera ord från SAOB, genom att använda av SAOB givna tidigare stavningsförekomster av bokstäver eller bokstavssekvenser?

Ja, det går bra att rekonstruera ord från SAOB, genom att använda av SAOB givna tidigare stavningsförekomster av bokstäver eller bokstavssekvenser (kap. 6.2.). 134 av de återskapade orden som påträffats med sökning av textord och kopplats till ordboksord finns i bilaga 4. Givet normaliseringsregler (Bilaga 1) har man en startmängd och en målmängd. Startmängden innehåller sekvenser som återger nutida stavning av fonem och målmängden sekvenser som återger nysvensk stavning av fonem. För ett ord med stavningsförekomst från SAOB tittar man i målmängden om det finns en kombination av regler som passar, väljer en sådan kombination och går via normaliseringsreglerna till motsvarande kombination i startmängden. Därefter substitueras den äldre stavningen in i ordet (kap 5.2). På detta sätt genereras ord. Vid mätning av sökningskavaliteten med en metod som tillåter ordböjning (kap. 6.5), först i enbart SAOB:s huvudord och variantord, därefter i SAOB:s huvudord och variantord med tillägg av de genererade orden, minskar precisionen från 0.9720 till 0.9535. *Recall* höjs ordentligt, från 0.7596 till 0.9486. F-score börjar på 0.8528 och blir sedan 0.9510. Detta sammantaget pekar i positiv riktning mot att det går att rekonstruera ord från SAOB, givet stavningsinformation. En lista med genererade ord som hittats med facitorden finns i bilaga 4.

Vilka regler gömmer sig bakom stavningsinformationen?

Man kan se det som att det är kombinationer av normaliseringsregler och identitetsregeln som ligger bakom stavningsinformationen (kap. 5.2).

Hur mäter man sökningskvaliteten?

Man har en mängd sökord som plockats från äldre texter. Dessa söker man med i en lista av SAOB:s ord, stavningsvarianter och genererade ord. Det är bra med två listor av godkända ändelser, en för sökorden och en för ordboksorden (bilaga. 3). Sedan kan man jämföra orden. Där orden skiljer sig, delas orden. Därefter tittar man i de godkända listorna för ändelser. Finns de avdelade sekvenserna i ändelselistorna är det en träff. Kvaliteten mäts sedan i antalet sant hittade ord, falskt hittade ord, sant ej hittade ord och falskt ej hittade ord. Jag konstaterar att det inte är relevant att mäta med godkända avledningar (kap. 5.5.1 & kap. 6.5) i nuläget. Eftersom orden jag genererar nya ord från är substantiv, kan inte avledda

ord från andra ordklasser bildas. Inte heller exakt matchning (kap. 5.5.3 & kap. 6.5) är intressant, då flera sökord är böjda och då inte skulle få träff med ett oböjt ordboksord. Den metod som däremot fungerar bra är att tillåta ordböjningar (kap 5.5.2 & kap. 6.5).

Litteraturlista:

Adesam, Y., Andersson, P., Borin, L., & Bouma, G. (2021). *A lexical resource for computational historical linguistics—Gothenburg university publications*. https://gup.ub.gu.se/publication/310933?lang=en

Bergman, G. (1984). *Kortfattad svensk språkhistoria* ([Ny utg.]). Prisma.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python* (1st ed). O'Reilly.

Bollmann, M., Petran, F., & Dipper, S. (2011). Rule-Based Normalization of Historical Texts. *Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, 34–42. https://aclanthology.org/W11-4106

Fornsvenska textbanken. (u.å.). *1734 års lag: Giftermålsbalk*. Hämtad 12 maj 2023, från https://project2.sol.lu.se/fornsvenska/

Huldén, M. (2009). *Foma—A Finite State Compiler and Library*. 29–32. https://fomafst.github.io/

Jurafsky, D., Martin, J. H., Norvig, P., & Russell, S. J. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (Second Edition, Pearson International Edition). Prentice Hall, Pearson Education International.

Litteraturbanken. (u.å.). *Samlade Verk 18. Amorina (1839) sida 5 etex | Litteraturbanken*. Hämtad 12 maj 2023, från https://litteraturbanken.se/f%C3%B6rfattare/AlmqvistCJL/titlar/SamladeVerk18/sida/5/etex

Pettersson, E., Megyesi, B., & Nivre, J. (2012a). Parsing the Past—Identification of Verb Constructions in Historical Text. *Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 65–74. https://aclanthology.org/W12-1010

Pettersson, E., Megyesi, B., & Nivre, J. (2012b). *Rule-based normalisation of historical text—A diachronic study*. 333–341.

https://konvens.org/proceedings/2012/pdf/50_pettersson12w/

Pettersson, G. (2005). *Svenska språket under sjuhundra år: En historia om svenskan och dess utforskande* (2., [uppdaterade] uppl.). Studentlitteratur.

Språkbanken. (u.å.-a). *Dalin: Then Swänska Argus 1732-1734* | *Språkbanken Text*. Hämtad 12 maj 2023, från https://spraakbanken.gu.se/resurser/dalin-then-swaanska-argus-1732-1734

Språkbanken. (u.å.-b). *Index of /gerlof/OpenEDGeS*. Hämtad 12 maj 2023, från http://demo.spraakdata.gu.se/gerlof/OpenEDGeS/

Språkbanken. (u.å.-c). *Svensk fraktur 1626-1816* | *Språkbanken Text*. Hämtad 12 maj 2023, från https://spraakbanken.gu.se/resurser/svensk-fraktur-1626-1816

Språkbanken. (u.å.-d). *Svenska tidningar 1818-1870* | *Språkbanken Text*. Hämtad 12 maj 2023, från https://spraakbanken.gu.se/resurser/svenska-tidningar-1818-1870

Språkbanken. (u.å.-e). *Svenska tidningar 1871-1906* | *Språkbanken Text*. Hämtad 12 maj 2023, från https://spraakbanken.gu.se/resurser/svenska-tidningar-1871-1906

Teleman, U. (2019). *Svensk ortografihistoria: Från 1200-tal till 1700-tal*.

Bilaga 1. Normaliseringsregler i tabellform

Normaliseringsregler för att återskapa ord inom tidsintervallet: **k** betyder konsonant, **v** vokal, GP, Gertrud Pettersson, GB, Gösta Bergman, UT, Ulf Teleman, GVB, Nya testamentet i Gustav Wasas bibel, YVgL Yngre Västgötalagen, BU Heliga Birgittas uppenbarelser, CB, Codes Bureanus, Elk, *Een lijten kockebook* (okänd författare), JS, Jesper Swedberg, E, Ekeblad, A, Aurivillius, UH, Urban Hiärne. De grönmarkerade har jag lagt till i efterhand genom att jämföra SAOB:s huvudord och variantord.

1225-1375	**1375-1526**	**1526-1732**	**1732-1906**	**Regler totalt**
				Inskottskonsonanter och z & x som s
ml -> mbl, mr -> mbr, -mer -> -mber, mn -> mpn, mt -> mpt, nr-> ndr -nner -> -nder, -n -> -nder lr -> ldr, GP s.98 **t-inskott**, ls->lz, ns->nz,GP s.98 Eft. dental (n/t/l): ts ->tz, ns -> nz, ls -> lz, YVgL, UT s.15 ns->nz, ls -> lz, ks -> kx CB, UT s.20 (ts/)ds -> z (ks/)gs -> x CB, UT s. 21 ((ts/)ds ->z) VgL, GB s.40	mn -> mpn, lr -> ldr, nr ->ndr, nr -> ndir, mr -> mbr, mr -> mbir BU, UT s.27 -mm-/-m- -> -mb- GB s.75 ds->ds/dz/dzs, sd-> sd/zd/szd ts-> ts/tz/tzs, st-> st/zt/szt, ls->ls/lz/lzs, sl ->sl/zl/szl) BU, UT s.27 gs->gx/(ibland)x/(ibland)xs ds->z BU, UT s.27 -ssl- -> -tsl-, (-tsl-/ -ll- -> -tl-) GB s.75	mr->mbr, mn->mpn GVB, UT s.44 -mb->-m, mbl->ml, mbn->mn, mpn->mn JS, UT s.70 inskott av b och p kvar,ml -> mbl, mn -> mpn traditionell eftersläpande stavning GB s.106 ns->nz, ds->dz, ss/s->sz/zs GVB, UT s.44 -s- -> -sz- -ss- -> -sz- E., UT s.56	inget inskott av b eller p GP s.162 [regeln slutar ds -> dz], GP s.162	m/mm -> mb träældomb, mn->mpn/mbn ml->mbl tumblare mr -> mbr timbra -mer -> -mber mt -> mpt transumpt nr/nnr ->ndr tinder -nner -> -nder, -n -> -nder lr/llr -> ldr taldrick ns->nz tendenz ks -> kx/x tobax gs->gx/x/xs x->ks/xs taks, tillväxst ds -> z/dz/dzs/tz/tzss

				tizfördriv, tidzdag, trodzs, trotz, trotzss sd-> zd/szd ts-> tz/ttz/z tzar, trottz st -> zt/szt/tzs tapitzseri (st/ss -> tzs tapisseri) används ej z->tz ls->lz/lzs sl -> zl/szl tiszl ss/s->sz/zs tisztel, traszel, trängzsel
				d/t, g/k och ng,nk,gn
g -> gh, ÄVgl, GB s.40 g->gh, ng -> ngh CB, UT s.19 g -> g/gh, gg -> g/gg CB, UT s.20 ng -> ngh/g/gg/ghg, nk ->nk/k CB, UT s.20 gn -> gn/ghn CB, UT s.20	(((Medeltid t ->th, d -> dh, g -> gh, ng -> ngh, UT s.9))) t -> t/ (enstaka th) BU, UT s.24 d->d/dh BU, UT s.25 (((<th> felaktigt för /t/ t->th BU, UT s.25))) -**v**ng -> -**v**ng/-**v**ngh, -**v**ng- -> -**v**ng-/-**v**ngh-, BU, UT s.27	(Swedberg, ta bort onödiga h efter d och g) [slutar gälla d-> dh, g -> gh] 1703 års bibel, GB s.94 1703 gh -> g GB s.94 [slutar gälla rd -> rdh, d->dh] GVB. UT s.43 t-> t/(enstaka th) GVB, UT s.43 **v**g->**v**g/**v**gh GVB, UT s.44,	1734 lag [slutar i stort sett gälla g -> gh, d -> dh] (1734, UT s.77) [slutar gälla g -> gh, d -> dh] GP s.155 1734+Svedberg d- -> th- GP s. 163	d -> dh, tandh t -> th/dh thal, täpedhe g -> gh, talgh k -> g tragt k -> gh tagh g -> k rd -> rdh ng -> ngh/nggh/ngn/gn/gg/g/ghg teringh, terninggh, tridiungn, tygn

	(((h fritt som prydnad BU, UT s.25))) g->(enstaka g)/gh BU, UT s.27 gn -> gn/ ghn BU, UT s. 27 d -> d/dh, g -> g/gh, suffix d-> d/(ibland dh) KrLL, UT s.35 (((hyperkorrekt h, d->dh, t->th Kl 35))) svagtoniga k, (-gh- -> -k-) (-dh -> -t) (Sverike blev Sverighe) GB s.75 **(((-v**dh - > -**v** GB s.75,))) -fdh- -> -dh- & -ghdh- -> -dh-, GB s.75	d->d/(ibland)dh CJ, UT s.51 g->g/gh CJ 51, -t -> -t/-th CJ, UT s.51 d-> d/dh, g->g/gh, -d -> -dh Elk, UT s.54 Skrytstavning, -t- -> -th-, -t -> -th E, UT s.56 [slutar gälla g -> gh, d -> dh] S, UT s.58, mell. vok. el. ordslut, **v**dh**v** -> **v**d**v**, **v**gh**v** -> **v**g**v**, -dh -> -d, -gh -> -g A, UT s.61, [slutar gälla enigt JS g -> gh, d -> dh] JS, UT s.70 Behåll t och <th> i de pronomiella orden som i bibel (om pronomen) d -> t/th JS, UT s.70 1734 lag d- -> th- GP s.163		gn -> ghn/n nk -> gk tagke (används ej) (om pronomen) d -> t/th
				Geminering av konsonanter

vkkv ->**vkv**/**vkkv,** [ej ordslut **-vk -> -vkk**] GB s.40, **vkv** ->**vkv**/**vkkv,** [ej ordslut **-vk -> -vkk**], CB, UT, s.21 **vkkv** -> enstaka **vkv/vkkv, vkv** -> enstaka **vkkv/vkv** YVgL, s. 15 UT Prydnadsgeminering i ordslut, -f -> -ff, -l -> -ll YVgL, s. 15 UT dd->(ofta d)/dd CB, UT s.20		bet. enstav. ord, kort vok. [ej ck -> kk], **kvkk** -> enstaka **kvk** **kvk** -> **kvkk** GVB, UT s.45, Enstav. kort vok. [ej ck -> kk], kvkk -> enstaka **kvk** -> **kvkk**] Elk, UT s.55 UH inte geminera kort. vok. enstav. [ej **-vk** -> -**vkk]** UH, UT s.73	1734 lag kort bet. vokal följs av geminerad konsonant, även i enstavingar. [ej ck -> kk], **kvkk** -> enstaka **kvk** -> **kvkk,** [ej -m -> -mm] UT s.77	**k** -> **kk** truddellutt **kk** -> **k** tip ck -> kk tack -m -> -mm tömm f -> ff, l -> ll tournesoll dd -> d tody
				ld
-ll- -> -ll-/-ld- GB s.42		GVB ll/ld -> ll/ld GP s.160		ll -> ld, ld -> ll tusengylden
				svarabhaktivokal
svarabhaktivokal, -**k**r- -> -**k**er-/-**k**ær-/-**k a**r-/-**ki**r- GB s.41 -**k**l -> -**k**el, -**k**n ->-**k**en GB s.41 YVgL	svarabhaktivokal i (enstaka e eller a) BU 24 svarabhaktivokal e Kl 35			

Svarabhaktivokal <e> ibland <i> UT 13 Svarabhaktivokalen, normalt a CB 19				
				Progressivt i-omljud
Progressivt i-omljud, -io- -> iō -io -> iø, GB s.40 -ia- -> -iæ-, GB s.40 CB <ia> växlar med <iæ> CB 18				
				i/j och u/v
1200-talet v initialt, u-/v- -> v-, u medialt, -u-/-v- ->-u- UT 8 YVgL i/j->i/j, u/v->u/v YVgL, UT s.13 u-/v- -> (ibland) u-/v-/w- CB, UT s.18	initialt v,u-/v- -> v-, medialt u -u-/-v- -> -u- BU, UT s.24 initialt j, i-/j- -> j-, medialt i, -i-/-j- -> -i- BU, UT s.24	1500-talet v konsonant, u vokal, j konsonant , i vokal [ej v -> u eller u -> v] UT 8 latinsk tradition u-/v- -> u-, -u-/-v- -> -u- CJ, UT s.51 v ->w, [ej v -> u eller u -> v] u vok., w kons. A, UT s.60 [ej i -> j eller j -> i]	Svedberg Init. v, u-/v- -> v- GP s.163 Santesson u- -> v-/u- GP s.163 1734 lag konservativt hi, si, ni, di istf. hj, sj, nj, dj **k**j->**k**i UT s.78	u/v -> u/v/w twg, tuaga, trvmme i/j -> i/j tjds-, täliare, troia **k**j -> **k**i tiog kius-

		i vok., j kons. A, UT s.60 (((/u/: u- -> v- JS, UT s.70)))		
				/v/
v -> f GB s.40 v- -> u-/v-/w-, **k**v- -> **k**u-, -**v**v**v**- ->- **v**u**v**-/-**v**fu**v-, -v**v**r-/-v**v**l- -> -v**u**r-/-v**u**l-, **v**v**- -> -v**f**/-v**ff, **-k**v **->-k**f YVgL, UT 14 v- -> u-/v-, kv- -> **k**v-/**k**w-/**k**u-, -**v**v**v- -> -v**u**v-/-v**f**v-/-v**v**v-/-v**w**v-,** -v -> -f CB, UT 20	v**v-** -> w**v**-/v**v**-/u**v**-, **k**v**v-** -> **k**w**v-/k**u**v-, -v**v**v- -> -v**f**w**v-/-v**w**v-, -v**k**v**v- -> -v**k**f**w**v-/-v**k**u**v-, -v**v**k**v- -> -v**f**k**v-, -**v **->** -f/-ff BU, UT 29 u->w, v->w Kl 35	/v/: init.<w> init. eft. kons. <w>, <u> med. efter vok.:ffu, ff med. eft. kons.: ffu fin. eft.vok. ff fin. eft. kons.: ff v- -> w-, **k**v- -> **k**w-/**k**u-, -**v**v- -> -**v**ffu-/-**v**ff-, -**k**v- -> -**k**ffu-, -v -> ff GVB, UT s.44 /v/ <w>, <sw> <fw> <fw> <ff> <ff> dubbleras framför <w> -> <ffw> CJ, UT 51 /v/ som GVB enstaka -v- ->-fu- medialt Elk, UT 54 1650 medialt /v/ -v- -> -fw- S, UT 57 [ej v->fw/fu], v -> w A, UT 62 enstav. -f -> -v A, UT 63	v- -> w-, sv- -> sw-, qv- -> qw-, hv- -> hw-, tv- -> tw-, -v- -> -fw-, -v -> -f 1734 lag, (UT s.77) 1800-t [ej -v- -> -w- -v-, -v -> -w, v- -> w-] GB s.160 1906 [ej -v -> -f, v- -> hv-, -v- -> -fv-] GB s.186 1906 [ej -v -> -f, v- -> hv-, -v- -> -fv-] GP s. 167	v -> hv/fv/fw/w/fu/ffu/ffw/ffv/ff/f torffwa, tafla tiwffw- sv- -> sw-, qv- -> qw-, hv- -> hw-, tv -> tw

		/v/ skall stavas f finalt och fw medialt Behåll hw -v -> -f, -v- -> -fw-, -v -> -w JS, UT s.70 ((()))Stava /v/ med w v->w, /hw/ med hw, medialt och finalt f får inte gemineras, /u/, u-> v/u UH, UT 74 1703 bibel [ej -v- -> -ffu-], -v- -> -fw- GB s.94 hv- -> fw-, ((()))GB s.105		
				Hiatus
hiatus -oa- -> -o- -ea- -> -e-, -oi- -> -o- (Svea rike, Sverige) GB s.41				
				h-bortfall
		hj- -> j-, dj- -> j, lj- -> j- GB s.105		hj- -> j-, dj- -> j, lj- -> j-
				Franska lånord
		k->c latinska lånord GVB 43	1800-t [ej k- ->vissa c-] GB s.159	k- -> c- -s -> -ce tolerance z -> ce tacett ce -> s

			(((-e -> -, GB s.159))) Denna regel används ej -ce -> -s GB s.159 -ll- -> -lj- GB s.159 -ch -> -sch GB s.159 franskt -ai- -> -ä-/-e-GB s.159 -eu- -> -ö- GB s.159 -ou- -> -u- GB s.159 -u- -> -y-/-u- GB s.159 -e- -> -ä-/-e- GB s.159	tasett (((s -> se))) lj ->ll tirailleur -sch -> -ch ä->ai ö -> eu trancheur eu -> ö taljör u -> ou -bousch o->ou tourner y -> u ä -> e eau->ou tombou å->eau trumeau eau->å trumå
				dt
			1800-t -t/-tt -> dt GB s.158 1900-t -dt- -> -t-/ -tt- GB s.186	t/d/tt->dt tandt, täcknadt, dt ->tt trått

				å/aa
	1400-talet, å -> a/aa/ (ibland) å UT 9 I BU fanns ej aa i form av /å/ [ej å -> aa] (Kl, UT 34) Nytt /å/ -> aa å -> aa Kl, UT 34 å- -> a-, ång -> ang, ård -> ard, åld -> ald GP s.148 å- -> a-, (årdh -> ardh) ård -> ardh, ång -> ang, åld -> ald, ånd -> and, åmb -> amb, ånk -> ank GB s.73 1300-t senare del kort /a/->/å/ GP s.148	1526 å införs [ej å->a/aa] 156 GP å används i NT 1526 och slog igenom [ej å->a/aa] UT 9 å användes för långt /o/ [ej å->a/aa] UT s.41		å -> (a/)aa (ång -> ang, ård -> ard, åld -> ald) (ånd -> and, åmb -> amb, ånk -> ank)
				kv/qu
YVgL kv -> qu UT 14 kv -> qu CB 19	kv->qu BU 24	kv->qu GVB 43 Behåll q i qw JS 70: kv->qw	qu för /kv/ kvar till långt in på 1800-t, kv -> qu UT s.9 Leopold kv -> qv s. GB s.158 SAOL -qv- ->-kv-/-qv- GB s.160	kv -> qu/qv/qw -k/-c/-q ->-que tiriaque k->qu/qv turniquett, trafiqvant qu -> k tournikett

			SAOL 1889 [ej kv -> qv] GP s.166 SAOL utgåvan 1900 kv obligatoriskt [ej kv -> qu] s. GB s.160	
				Suffix
YVgL suffix i -> e, e -> i,u -> o, o -> u UT 13 YVgL suffix a -> æ, æ->a UT 13 OBS ej i CB! suffix, ibland o -> u CB 18 suffix i -> e, e -> i,u -< o, o -> u CB 18 vokalbalans, kortstavigt ord, a/i/u i ändelsen, långstavigt ord ä/e/o i ändelsen GB s.42 ändelsevok. a, u, i GP 96 best. art. e/i CB 19	suffix i -> e, e -> i,u -< o, o -> u BU 23	svagtoniga suffix (e,a,o), enstaka fall u och i GVB 42 a, i ändelser som i daniserad svenska har a GVB 42 suffix e och o, enst undantag u, -ig(h), -lig(h), -ing CJ 50 Ändelsevokal, best. art. e Elk 54	Salvius -ligit->-igt GP s.163	
				Affricering
YVgL inkonsekvent	Pal./Affr. vid k, g utmärks ej	Ingen palatalisering av kä, gö [ej	-ga -> -gia, -ka -> -kia GB s.137	kö -> kiö kiörne

kæ -> kiæ, gæ ->giæ YVgL, UT s.14 (CB <ia> växlar med <iæ> CB, UT 18 Skall igentligen inte ligga under denna rubrik) kæ -> kiæ, gæ ->giæ CB, UT s.19	[ej kä ->kiæ, kö -> kiö , gä -> gie, gö -> giö] BU, UT s.25	kä ->kiæ, kö -> kiö, ke -> kie , gö -> giö, ge -> gie]. GVB, UT s.43 [ej kä ->kiæ, kö -> kiö, gä -> giæ] NT, UT s.66' kö->kiö, gö->giö JS, UT s.70 [ej gä -> giä, kö -> kiö, kä -> kiä] UH, UT s.74	Salvius: [ej ke-> kie, kä -> kiä, gö -> giö] GP s.163 1734 lag /g/, /k/, /sk/ ej konsekvent affricerats kö -> kö/kiö, kä -> kä/kiä, ke -> kie, gö -> giö, ge -> gie, gä -> giä UT, s.77	kä -> kiä/kjä/kiæ/kie kierna kiärna, kiära ke -> kie teckie gö -> giö gä -> gie/giä/giæ ge -> gie ö -> iö tusenskiöna tillagd genom obseration av stavningsinformation -iö-
				ft/pt och f/ph
YVgL ft- > pt/ibland ft UT 14 ft -> pt CB 19	ft -> pt BU 25 fft -> f GB s.94	Inget fft, GP s. 162		ft -> pt tapt f -> ph trumph ft -> fft tålfft ff->ffu toffuel
				Geminering av vokaler
-vk -> -**vk**/-**vvk,** -**v** -> -**v**/ enstaka -**vv** CB, UT s.21	Enstavingar, -**vk** -> -**vk**/-**vvk,** -**v** -> -**v**/-**vv** UT 10 -**v-** ->-**vv-,** -**v** -> -**v**/(ibland) -**vv** BU, UT s.33 i->ij/ sällan ii BU, UT s.33	Geminerat i, i-> ij GVB, UT s.42 Lång vok i enstavingar**, även i enstaviga ords slut** -**vk** ->-**vk**/-**vvk,** -**v** -> -**v**/-**vv** GVB, UT s.44 lång vokal dubbelskrevs i	[ej **v**->**vv** , i->ij] GP s.162 1734 lag lång vokal gemineras aldrig, [ej **v**->**vv**] UT, s.77	**v** -> **vv** i -> ij, ii->ij tijm ii -> ijij tijijm

		book, men enkelskrevs i öppen stavelse boken, dessutom framför dh, gh, ff, dödh, dagh, aff, [ej **v**dh -> **vv**dh, **v**gh -> **vv**gh, **v**ff-> **vv**ff], (för konsonanter utom digraferna dh, gh och ff) **kvk**-> **kvvk** Bibeln 1541 GB s.91 (((lång vok. **v**->**vv/vh** CJ, UT s.52))) -**kvk**- ->-**kvk**-/-**kvvk**-, -**kv** -> -**kv**/-**kvv** Elk, UT s.55 **v** ->**v**/ sällan **vv**, ij->i A UT s.62 [ej **v** -> **vv**, i->ij], JS, UT s.68		
				w som långt u
geminerat u u -> w UT 10		långt u -> w, BU, UT 23 långt och kort u->w, (KrLL, UT s.35)		u -> w tjwg (uu->w) treuuffes SAOB används ej

		[ej **k**u**k**->**k**w**k**/**k**uu**k**] JS, UT 68 1703 bibel [ej -uu- -> -w-] GB s.95 långt u ->ibland w GVB, UT 42 långt /u:/ i enstav. u->w, CJ, s.51, UT [ej u -> w] A, UT s.62		(-w- -> -uu-) används ej
				Lång vokal inskotts h
	(((**v** -> **v**h, UT 9)))När??	lång vok. **v**->**v**/**vv**/**v**h CJ, UT s.52 få ord, lång **v** **v**->**v**/ få **v**h Elk, UT s.55 Avvisar lång vok. [ej **v**->**v**h], (från exempelord) [ej **v**r->**v**hr] A, UT s.62	1734+Svedberg lång vok. **v**k->**v**h**k** GP s. 163 Salvius [ej **v**k -> **v**h**k**, **v**r->**v**hr] GP s.163 1734 lag lång vokal ibland med h **v**l -> **v**hl, **v**r -> **v**hr, **v**m -> **v**hm, **v**n -> **v**hn UT, s.77	**v** -> **v**h **v**r -> **v**hr tanåhr **v**l -> **v**hl tahl **v**m -> **v**hm töhm **v**n -> **v**hn teehn
				y/i
y->i, i-> y CB 18	i -> i/(enstaka y) BU 23 y -> y/(enstaka yu)´BU 23 -i- -> -e- -y- -> -ø- GB s.73	y-> y/i GVB 42		i->y/yu

				/k/
YVgL k->k/c, UT 14 k->k/c, k -> (enstaka) ch, nk -> nck, ck->kk CB, UT s.19	ck -> kk, BU, UT s.24 k->k/c/ch, särskilt sk->sk/sc/sch, ck -> kk/(ibland ck) Kl, UT s.36	(((k -> k framför palatal vokal, annars k -> c UT 9))) (((k -> kk, k -> ck UT 9))) k -> k/enstaka c/ck, [ej ck -> kk], lk->lck, rk -> rck, nk ->nck, kt -> cht GVB, UTs. 43 /k/ tecknas ck efter n, l och r lk->lck, rk -> rck, nk ->nck, kt->cht, [ej ck -> kk] CJ, UT 51 /kt/ kt->cht, [ej ck -> kk] Elk, UT s.54 Behåll cht, kt -> cht JS, UT s.70	1734 lk->lck, nk->nck, rk->rck, kt -> cht/kt/gt/ckt GP s.163 1734 lk->lck, nk->nck, rk->rck, kt -> cht/kt/gt/ckt GP s.163 1734 lag kt->cht UT, s.78 Salvius tryckeri [ej kt/gt -> cht/ckt] GP s. 163 1800-t Franskt c byttes ut mot k s. [ej k- ->vissa c-] GB s.159	k -> c ck -> kk k -> q trafiq k -> ch ck -> ch techie SAOB kt -> gt/cht/ckt lk ->lck, nk -> nck, tanck rk -> rck sk ->sc/sch/gsk/tsk/sch k/sck trocisc, tascha, tredschk-, tresck
				æ/ä/e, ø/ö och å/o
YVgL kort bet. vok. ä -> e/æ YVgL, UT s.13 ä -> e/ oftast æ CB, UT s.18	ä -> e/ oftast æ BU, UT s.23 bibeltryck 1500-t æ blir ä, ø blir ö [ej ä ->æ , ö -> ø] GP s.156	ä -> e/ä GVB, UT s.42 kort /o/: å/o-> å/o CJ 50 Kort betonat /ɛ/: e/ä -> e/ä CJ, UT s.50 /jä/: jä->ie CJ, UT s.50 kort vok. e och o Js, UT s.69	1734 lag ä -> ä/e UT s.78 SAOL 6 1889 [ej är -> er, jä -> je] ((())) GP s.166	ä -> e/æ e -> ä ö -> ø å/o -> å/o ge -> je är -> er jä ->ie/ je ga -> je

		å -> o/å ä -> e/ä, GP s. 160		ß -> ss y ->ÿ/ iÿ kÿf ky -> kiy/kiÿ kiiyf ti -> tiy
				ch, sh, tj, sch och ge
				ge - >sch/sie trikåtasch sh ->sch/ch -bosch tj ->sch/ch/tch/cs/tsch/ tsj/tschj/ti/kj/ki/k/c/cz /z tscherkess tiur, kjuder, kiäll, kär- cyrkasch, czech zerkass tsch -> tsj tsjibuk sch -> sh/sj/sk/ch/che/rs/sk j tush, tusj, tusk, touch tartche mars skjack sk -> sch/schk tusenschöna, trocisch ch -> k/sch/si transchement siåklad

				che -> ch sj -> ss tjuvass rs ->rts/rtz/rse/rce/sch/ sche/rsche/tz/tj werts, tertz, tarse, tesche tersche tertja rt -> rs/rc tersial, tercial stj -> sj/sk/sti/st sjern, skern, stiern stielk (((tion -> ? hittar ingen variant)))
				g -> j SAOB (gikt/jicht)

Bilaga 2. Språkhistorien bakom normaliseringsreglerna i kronologisk ordning

Inskottskonsonanterna d, b, p, t samt användning av z och x:

k betyder konsonant, **v** vokal, GP, Gertrud Pettersson, GB, Gösta Bergman, UT, Ulf Teleman, GVB, Nya testamentet i Gustav Wasas bibel, YVgL Yngre Västgötalagen, BU Heliga Birgittas uppenbarelser, CB, Codes Bureanus, Elk, *Een lijten kockebook* (okänd författare), JS, Jesper Swedberg, E, Ekeblad, A, Aurivillius, UH, Urban Hiärne

- Inskottskonsonanter förekom ända från runsvenskan (GB s.20) till 1734 års lag (humblegård) (GB s.106).
- Runsvenska: Mellan *ll* och *r* samt mellan *nn* och *r* sköts ett *d* in. Mellan *m* och *l* samt mellan *m* och *r* sköts ett *b* in. Mellan *m* och *n* samt mellan *m* och *t* sköts ett *p* in (GB s.25). Detta ger upphov till reglerna -llr- -> -ldr-, -nnr- -> -ndr-, -ml- ->-mbl-, -mr- -> -mbr-, -mn- -> -mpn-, -mt- -> -mpt-,
- Fornsvenska: I text ser vi även *t*-inskott mellan *l* och *s* samt mellan *n* och *s* (*manz*), *b*-inskott mellan m och l samt mellan *m* och *r* kvarstår så även *p*-inskott mellan *m* och *n* samt mellan *m* och *t* (*nampn* ‘namn’). *d*-inskott är nu mellan *l* och *r* samt mellan *n* och *r* (*ormber* ‘orm’, *munder* ‘mun’) (GP s.98).
- Vi har nu reglerna ml -> mbl, mr -> mbr, mn -> mpn, mt -> mpt (ormer->ormber eller orm -> ormber, -m -> -mber), lr -> ldr, nr-> ndr, (munner -> munder, -nner -> -nder eller mun -> munder, -n -> -nder), (t-inskott) ls->lz, ns->nz
- *s* varierar med *z* efter dentalerna *t*, *n*, *l* (manz) (YVgL, UT s.15): ts ->tz, ns -> nz, ls -> lz
- /s/ skrivs med *x* efter palatalt/velart ljud (CB, UT s. 20): ks -> kx
- *z* och *x* kan stå i konsonantförbindelser som i latinet (CB, UT s. 21): *z* för /ts/ (guz) och *x* för *ks* (dax): ts/ds -> z, ks/gs -> x
- I Äldre västgötalagen uttalades *z* som *ts* (uplænzk) (VgL, GB s.40): ts/ds ->z.

1375-1526:

- Inskottskonsonanter kvar enligt ovan, även -nr -> -ndir, -mr -> -mbir (BU, UT s.27): mn -> mpn, lr -> ldr, nr ->ndr, nr -> ndir, mr -> mbr, mr -> mbir
- *s* ersätts eller kompletteras med *z* intill dental (BU, UT s.27): ds->ds/dz/dzs, sd-> sd/zd/szd ts-> ts/tz/tzs, st-> st/zt/szt, ls->ls/lz/lzs, sl ->sl/zl/szl).
- *s* intill palatal/velar ersätts ibland med *x*: gs->gx.
- z och x används ibland som i latinet för *s* intill dental respektive palatal/velar (BU UT s.27): ds -> z, gs->gx/(ibland)x/(ibland)xs
- Mellan *t* och *l* utvecklades ett *s*, av *vatle* blev *vatsle* sedan blev *vatsle vassle*, utom i sydsvenska där *tl* assimilerats till *ll* som i danskan. I götamål står *tl* kvar (GB s.75): -ssl- -> -tsl-, (-tsl-/ -ll- -> -tl-)
- Assimilation av *mb* har gett *lamm*, *kam* och *dimma* av *lamb*, *kamb* och *dimba* (GB s.75): -mm-/-m- -> -mb- , -mm/-m -> -mb

1526-1732:

- Inskott av *b* mellan *m* och *r* samt av *p* mellan *m* och *n* är kvar (GVB, UT s.44): mr->mbr, mn->mpn
- Swedberg vill slopa onödiga *b* och *p* i ord som *kamb*, *himblar* och *nampn* (JS, UT s.70):-mb->-m, mbl->ml, mbn->mn, mpn->mn

- Traditionell eftersläpande stavning med inskott av *b* och *p* hos Horn, Ekeblad, Columbus, Karl XI och Karl XII (*gambla*, *nampn*) (GB s.106): ml -> mbl, mn -> mpn
- *s* ersätts ibland med *z* efter dental (*wingårdenz*, *manz*, *gudz*), istället för geminerat *ss* används ibland som i tyskan *sz* eller *zs* (*gniszlar*, *korszet*) (GVB, UT s.44): ns->nz, ds->dz, ss/s->sz/zs
- Ekeblad använder medialt *sz* för *s* oavsett om det är geminerat eller ej (E, UT s.56): -s- -> -sz-, -ss- -> -sz-

1732-1906

- Några decennier in på 1700-talet förekommer inte längre inskott av *b* eller *p* mellan *m* och *l*, *r*, *n* eller *t*, inte heller förekommer *z* som tecken för *z* (*Guds*, *lands*) enligt Santesson (1986 s. 276 f.) (GP s.162): Föregående regel gäller ej längre [regeln slutar ds -> dz]

þ och đ

1225-1375:

- þ hade övertagits från runskriften för frikativ dental, både tonande och tonlös, som t*h* i engelska. VgL, (GB s.40): (dh/th -> þ)
- *þ* och *đ* försvinner. Svenskan under 1300-talet bestod enbart av latinska bokstäver (UT s.11).
- [ej th -> þ, dh -> đ]
- Ortografisk variation mellan *đ* och *d* (*đotter*, *næmpdæmannæ*, dotter) (YVgL, UT s.14): d -> d/đ
- Ofta enkelt *d* i stället för gemination i kontrast mot *þ*. (*klæda* ‘klädda’ mot *klæþa* ‘kläda’) (CB, UT s.20): dd ->d/þ
- */þ/* vanligen *þ*, ibland *th* (YVgL, UT s.14): th- -> þ-, -dh-/(ibland)-th- -> -þ-, -dh/(ibland)-th -> -þ,
- *þ* används fortfarande (*þola*) även för */đ/* (var), alternativt användes *th* (*hughnath*) (CB, UT s.19): (đ -> þ, þ ->þ/th)

1375-1526:

- */þ/* hade fallit i uttalet och ersatts av *th* (BU, UT s.24): [ej th -> þ]
- I BU betecknades äldre */d/* med *d*, *dd* och *ddh*, äldre */đ/* med *dh* (BU, UT s.26): d -> d/dd/ddh, [ej]dh -> đ]
- th användes istf. */þ /* i början av ord (*thingh*, *thört*, *thola*, *thänkiande*), samt felaktigt för äldre */t/* (*luth*, *mathin*) (BU, UT s.25): [ej th- -> þ-] , t -> th
- Bokstaven *þ* förekommer fram till ca 1400. Det står för både tonande och tonlös frikativa (GP s.59).
- Bokstaven *þ* försvinner i början av den yngre fornsvenska perioden, Man skriver *th* i början av ord, annars *dh* (GB s.74): [ej th- -> þ-, -dh- -> -þ-, -dh -> -þ]

d/t, g/k och ng

1225-1375

- I faksimilet av ÄVgL var *gh* tecknet för frikativt *g* (GB s.40): g -> gh
- I CB hittar vi gh och ngh (CB, UT s.19): g->gh, ng -> ngh
- *g* hade ljudvärde */g/* initialt eller geminerat *g* (*giuæ*, skugge). Spirantiskt */ʒ/* skrivs *gh* (*dagh*). *Geminerat g* kunde även betecknas med enkelt *g* (*skuge*) (CB, UT s.20): g -> g/gh, gg -> g/gg
- */ng/* kan också skrivas utan nasalbokstav framför *g* (*siugga*, *siughga* ‘sjunga’) eller framför *k* (*skækto* ‘skænkto’) (CB, UT s.20): ng -> ngh/g/gg/ghg, nk ->nk/k
- Uttalet av *gn* osäkert (dygn, rægn, vaghn) (CB, UT s.20): gn -> gn/ghn

1375-1526

- */t/* stavas med *t* i BU, någon enstaka gång med *th* (BU, UT s.24) t -> t/ (enstaka th).
- Tonande spirant markeras med d eller dh BU, UT s.25: d->d/dh
- ((th användes också felaktigt för gamla /t/ (UT s.25: t -> th)))
- ng och ibland ngh förekommer finalt och medialt efter vokal (BU, UT s.27): -**v**ng -> -**v**ng/-**v**ngh, -**v**ng- -> -**v**ng-/-**v**ngh-
- g eller gh står sannolikt för nasal i (vägna wäghna, sägn)(BU, UT s.27): gn -> gn/ghn
- /g/ betecknas normalt med gh, i enstaka fall med g (BU, UT s.27): g->(enstaka g)/gh
- dh och gh blandas med d och g (laghman, lagmen), i suffix föredras d (skilnader)(KrLL, UT s.35): d -> d/dh, g -> g/gh, suffix d-> d/(ibland dh)
- Yngre fornsvenska: Svagtonigt k blev gh, Sverike blev Sverighe och t i obetonad ställning blev dh (mykit blev mykith) (GB s.75): (-gh- -> -k-) (-dh -> -t)

1526 - 1732

- Swedberg i den språkkommission som stod för ändringarna i 1703 års bibel tyckte att onödiga h efter d och g skulle bort (GB s.94) [slutar gälla d-> dh, g -> gh]
- I GVB skrivs d efte r, inte dh (ord) (GVB. UT s.43): [slutar gälla rd -> rdh, d->dh]
- /t/ har normalt betckningen t i GVB, endast i enstaka ord th (vth) (GVB, UT s.43): t -> t/(enstaka th)
- /g/ betecknas g utom för spirantiskt uttal efter vokal, för då betecknas det gh (GVB, UT s.44): **v**g->**v**g/**v**gh
- Både g och gh förekommer i CJ, så även t och ibland th i ordslut, samt d och ibland dh (CJ, UT s.51): d -> d/(ibland)dh, g->g/gh, -t -> -t/-th
- Spiranterna kan vara på väg att försvinna, ty g växlar med gh, så även d med dh, dock föredras dh i ordslut (Elk, UT s.54): d-> d/dh, g->g/gh, -d -> -dh
- Ekeblad har inslag av skrytstavning. E skriver th istället för t i icke-initial position:(UT s.56): -t- -> -th-, -t -> -th
- 1600-talets mitt: Santesson(1988) har visat att dh och gh har utgått (S, UT s.58): [slutar gälla g -> gh, d -> dh]
- Aurivillius undviker dh och gh mellan vokaler och i ordslut (A, UT s.61): **v**dh**v** -> **v**d**v**, **v**gh**v** -> **v**g**v**, -dh -> -d, -gh -> -g
- JS tycker också att man skall skriva d och g istället för dh och gh (S, UT s.70):
- [slutar gälla enigt JS g -> gh, d -> dh]
- Swedberg ville behålla t och th i de pronomiella orden som i bibelspråket
- JS, (UT s.70): (om pronomen) d -> t/th

1732-1906:

- I 1734 års lag förekom initialt th gentemot d (then, ther) (GP s.163): d- -> th-
- dh och gh förekommer nästan aldrig i 1734 års lag (1734, UT s.77): [slutar i stort sett gälla g -> gh, d -> dh]
- I början av 1700-talet har dh och gh försvunnit (GP s.155): [slutar gälla g -> gh, d -> dh]

Geminering av konsonanter

1225-1375:

- I faksimilet av ÄVgL dubbelskrevs långa konsonanter mellan vokaler (sunnan, allum), men inte i ordslut (ræt) (GB s.40): **vkkv** ->**vkv**/**vkkv,** [ej **-vk -> -vkk**]
- I CB förekommer ej geminering av slutkonsonant (säk), men av lång konsonant efter kort betonad vokal (UT s.21) annars: **vkkv** ->**vkv**/**vkkv,** [ej **-vk -> -vkk**]
- I YVgL finns geminering av lång konsonant efter betonad kort vokal (ættarbot) i enstaka fall (driker), undantagsvis även efter beton. lång vok (YVgL, s. 15 UT): **vkkv** -> enstaka **vkv/vkkv, vkv** -> enstaka **vkkv/vkv**
- Ibland prydnadsgeminering av f och l i ordslut (YVgL, UT s. 15): -l -> -ll, -f -> -ff
- Ofta användes enkelt d istället för gemination (CB, UT s.20): dd->(ofta d)/dd

1526 - 1732:

- Konsonanttecknet dubbleras ofta I betonade enstaviga ord på konsonant om vokalen är kort (gick, klädd, nytt) (GVB, UT s.45). Som synes även i ordslut, dock ck istället för kk.:
- [ej ck -> kk], kvkk -> enstaka **kvk** -> **kvkk**
- Även i Elk dubbleras konsonanttecknet i betonade enstavingar på konsonant när vokalen är kort (hull, fett, kött, tryck), endast undantagsvis sker ingen geminering (skin, prop) (Elk, UT s.55). ck skrivs istället för kk: [ej ck -> kk], kvkk -> enstaka **kvk** -> **kvkk**]
- Urban Hjärne ville inte geminera slutkonsonanten i enstaviga ord (UH, UT s.73): [ej **-vk** -> -**vkk]**

1732-1906:

- I 1734 års lag gemineras aldrig ett finalt m (söm, ström). De andra konsonanterna gemineras efter kort betonad vokal, även i enstavingar, utom k som geminerat skrivs ck, undantag utgör frekventa strukturord (at, skal, wil, til, up) (UT, s.77) [ej ck -> kk], **kvkk** -> enstaka **kvk** -> **kvkk,** [ej -m -> -mm]

ld

1225-1375:

- Under klassisk fornsvensk tid övergick ld till ll (GB s.42):-ll- -> -ll-/-ld-
- I GVB växlar ld med ll (GP s.160): ll/ld -> ll/ld

i/j och u/v

- Något som inte längre förekommer är fenomenet att i växlar med j, både som vokal och som konsonant, inte heller förekommer det att u växlar med v, både som vokal och som konsonant.

1225-1375

- Svenskan upptog den latinska ortografin på 1200-talet, där både u och v kunde stå för vokal respektive konsonant.v användes initialt och u medialt (UT s.8): u-/v- -> v-, -u-/-v- ->-u-
- Paren i och j samt u och v växlar i YVgL . De är grafiska varianter och kan alla stå för både vokal och konsonant (YVgL, UT s.13): i/j->i/j, u/v->u/v
- I CB växlar u med v, så att v eller w står initialt för både vokal och konsonant-ljud, ibland förekommer också u initialt (CB, UT s.18): u-/v- -> (ibland) u-/v-/w-

1375-1526

- I BU används v initialt för u/v och u medialt för u/v (BU, UT 24): u-/v- -> v-, -u-/-v- -> -u-
- j stod i BU initialt för i/j och i medialt för i/j (BU, UT s.24): i-/j- -> j-, -i-/-j- -> -i-

1526 - 1732:

- I CJ är den latinska traditionen med v initialt för u/v och u medialt för u/v kvar (CJ, UT s.51): u-/v- -> u-, -u-/-v- -> -u-
- Först på 1500-talet genom Petrus Ramus fördelades u till vokal och v till konsonant samt i till vokal och j till konsonant (UT s.8): [ej v -> u eller u -> v, i -> j eller j -> i]
- Aurivillius införde en uppdelning av i/j respektive u/v med i som vokal och j som konsonant, respektive u för vokal och w för konsonant (A, UT s.60): v ->w, [ej v -> u eller u -> v, i -> j eller j -> i]
- Swedberg inför ytterligare att w reserveras som konsonanttecken även i förbindelse med föregående konsonant (sw, hw, tw, qw), samt skriver i efter initial konsonant. Vokalen u skrivs v i början av ord, annars (JS, UT s.70): u- -> v-)))

1732-1906:

- På 1700 -talet gäller fortfarande att v kan vara ett vokalgrafem som sätts initialt för u. (Santesson, GP s.163): u- -> v-/u-
- 1734 års lag stavar konservativt (hionalagh,siö, niuta, diupt) (UT s.78): **k**j->**k**i

/v/

1225-1375:

- I ÄVgL var f också tecken för /v/ (GB s.40): v -> f
- I YVgL tecknas /v/ u,v eller w initialt, u efter initial konsonant, u eller fu medialt mellan vokaler, u medialt efter betonad vokal följd av likvida, f eller ff finalt efter vokal, samt f finalt efter konsonant (af, lof, aff, loff) (YVgL, UT s.14): v- -> u-/v-/w-, **k**v- -> **k**u-, -**v**v**v**- ->- **v**u**v**-/-**v**fu**v-, -v**v**r-/-v**v**l- -> -v**u**r-/-v**u**l-, v**v- -> -v**f/-v**ff**, -k**v **->-k**f
- I CB skrivs /v/ initialt u eller v, efter initial konsonant v,w,u, medialt mellan vokaler u/f/v/w och finalt -f (CB, UT 20): v- -> u-/v-, kv- -> **k**v-/**k**w-/**k**u-, -**v**v**v- -> -v**u**v-/-v**f**v-/-v**v**v-/-v**w**v-,** -v -> -f

1375-1526:

- I BU Initialt före vokal tecknades /v/ w, u eller v, initialt mellan konsonant och vokal w eller u, medialt mellan två vokaler w eller fw, medialt efter vokal och konsonsnt följt av vokal u eller fw, medialt efter vokal före konsonant och vokal f, finalt f eller ff (hawir, owir, grawa, arff, loff, uphoff, af, aff) (BU, UT s.29): v**v-** -> w**v**-/v**v**-/u**v**-, **k**v**v-** -> **k**w**v**-/**k**u**v-, -v**v**v- -> -v**f**w**v-/-v**w**v-, -v**k**v**v- -> -v**k**f**w**v-/-v**k**u**v-, -v**v**k**v- -> -v**f**k**v-, -**v **->** **-f**/-ff
- Skrivaren använder i KrLL w både för v, och kort respektive långt u (KrLL, UT s.35): u->w, v->w

1526 - 1732:

- I GVB tecknas /v/ initialt w, initialt efter konsonant w eller u, medialt efter vokal ffu eller ff, medialt efter konsonant ffu, finalt ff (wägh, watn) (GVB, UT s.44): v- -> w-, **k**v- -> **k**w-/**k**u-, -**v**v- -> -**v**ffu-/-**v**ff-, -**k**v- -> -**k**ffu-, -v -> ff
- I CJ stavas /v/ initialt w, initialt efter konsonant w, medialt fw/ffw, finalt ff (swars, sielfwa, haff, loff) (CJ, UT s.51): v- -> w-, **k**v- -> **k**w-/**k**u-, -v- -> -fw-/-ffw-, -v -> -ff (((**v** ock **k** framför med. och fin. eller utan?)))
- I Elk följer /v/ huvudprinciperna i GVB (wisp, hwijta, skifwor, korff, halff), men stavas med enstaka fu medialt. Endast prepositionen av skrivs både med enkelt och dubbelt f (Elk, UT s.54): enstaka -v- ->-fu- , av -> af/aff (((Tycker inte att texten stämmer, kolla noga, fw finns inte medialt i GVB OBS, flytta formler till tabell!)))
- Runt 1650 stramades reglerna upp. /v/ böjade stavas fw medialt (S, UT s.57): -v- -> -fw-
- A ansåg att /v/ skulle stavas w i alla positioner. Han ogillade fw och fu (A, UT s.62): [ej v->fw/fu], v -> w
- A ville inte heller ha f i enstavingar ordslut (A, UT s.63): [ej -v -> -f]
- Swedberg ville att /v/ stavas f finalt, fw medialt och w annars, behåll hw (JS, UT s.70):
- -v -> -f, -v- -> -fw-, v- -> w-
- Hjärne rekommenderar att /v/ stavas med w, han vill även behålla hw. medialt. Stavning av /v/ med f medialt och finalt skall inte gemineras (UH, UT s.74): ((()))
- I 1703 års bibel skrivs fw istället för ffu (GB s.94): [ej -v- -> ffu], -v- -> fw
- hv uttalades hw av Swedberg i “Schibboleth” (1716): hv- -> hw- ((()))

1732-1906:

- I 1734 års lag stavas /v/ initialt w (wäg), efter s,q,h,t w (oqwädingsord, swek, twägga), medialt fw och finalt f (UT s.77): v- -> w-, sv- -> sw-, qv- -> qw-, hv- -> hw-, tv- -> tw-, -v- -> -fw-, -v -> -f
- Ungefär 1876 ersattes w med v (GB s.160): [ej -v- -> -w- -v-, -v -> -w, v- -> w-]
- Fridtjuv Berg utfärdade 1906 att f, fv och hv skulle ersättas med v som /v/-ljud (GP s.167) (GB s.186): [ej -v -> -f, v- -> hv-, -v- -> -fv-]

Geminering av vokaler:

1225-1375:

- Betonad lång vokal med efterföljande konsonant förekommer finalt både med och utan geminering (tro, mö, do), i ordslut finns enstaka fall med geminering (lee, doo, bii) (CB, UT s.21): -**vk** -> -**vk**/-**vvk,**-**v** -> -**v**/ enstaka -**vv**

1375-1526:

- I yngre fornsvenska geminerades lång vokal i enstaviga ord (kaal ‘kål’), även då vokalen stod finalt (skee) (UT s.10): -**vk** -> -**vk**/-**vvk,** -**v** -> -**v**/-**vv**
- I BU dubbleras ofta lång vokal (been, skiin, (skin), liif) (BU, UT s.33), endast enstavingar i exempel. Ibland geminering av vokal i ordslut (BU, UT s.33): -**v-** ->-**vv-,** -**v** -> -**v**/(ibland) -**vv**
- Normalt stavades långt i ij i BU (BU, UT s.33): i->ij/ sällan ii

1526 - 1732:

- I GVB gemineras ofta lång vokal i enstavingar (syyn, håår, taal, näät) (GVB, UT s.45), ofta gemineras dessutom lång vokal i enstaviga ords slut (knää, hoo) (GVB, UT s.44): -**vk** ->-**vk**/-**vvk,** -**v** -> -**v**/-**vv**
- För geminerat i användes ij (GVB, UT s.44): i-> ij
- Lång vokal enkelskrevs i öppen stavelse (boken, hopen, saken) och framför dh, gh och ff (dödh, dagh, aff), men dubbelskrevs i book, hoop, saak, Bibeln 1541 (GB s.91): [ej **v**dh -> **vv**dh], **v**gh -> **vv**gh, **v**ff-> **vv**ff], (för konsonanter utom digraferna dh, gh och ff) **kvk**-> **kvvk**
- (((lång vok. **v**->**vv/vh**
- CJ, UT s.52)))
- I En liten koke gemineras vokaler lite varstans (hoopa), även i slutvokal blandas enkel och geminerad vokal (slåå, slå) (Elk, UT s.55): -**kvk**- ->-**kvk**-/-**kvvk**-, -**kv** -> -**kv**/-**kvv**
- Aurivillius ville undvika vokalgeminering (A UT s.62): **v** ->**v**/ sällan **vv**
- Swedberg ville inte alls ha dubblerade vokaler för vokallängd (JS, UT s.68): [ej **v** -> **vv**, i->ij]
- I 1703 års bibel dubbelskrevs vokaler som i 1541 års bibel (löön, meente, wijn) (GB s.94) (((tillför tabell)))

1732-1906:

- 1734 års lag innehöll ingen voklageminering, ij bannlyses också (UT, s.77): [ej **v**->**vv**]
- Lag 1734 innehöll endast undantagsvis dubbeltecknade vokaler
- Några decennier in på 1700-talet förekom inte dubbeltecknade vokaler och därför inte heller geminerat *i* i form av *ij* (GP 162): [ej **v**->**vv** , i->ij]

w som uu

1225-1375:

- 1400-t långt u skrevs w (mwr, wt, siwk, diwr) (BU, UT s.23): u->w

- 1400-t Kristoffers landslag, w för u oavsett kort eller långt u (wth, hws, siwke, twnga, wundher) (KrLL, UT s.35) (OOOBS! wu TILLISAMMANS I FÖREGÅENDE ORD!!!: u->w,

1526 - 1732

- I GVB stavades långt u med w ibland (GVB, UT s.42): långt u -> w
- Långt u återges ofta i enstavingar med w (liws, grws, siw) (CJ (1640), s.51, s.UT): u -> w,
- Aurivillius bestämde att w skulle vara konsonant, sålunda fick man inte stava hws (A, (UT s.62) [ej u -> w]
- I 1703 års bibel skrev man uu istället för w som tecken för u (GB s.95) (huusbondan): [ej -uu- -> -w-]
- Swedberg som inte vill ha dubbelvokaler, vill ej heller ha w för dessa (JS, UT s.68), vilket senare följdes i I Molins fickbibel 1720
- :[ej **k**u**k**->**k**w**k/k**uu**k**]

vh

1526 - 1732

- Lång vokal kan återges både med enkel och dubbel vokal, samt med ett efterföljande h (wååren, soot, leek, röök; kohl, åhr, ehoo, siöö, wåt, wedh, åå (subst.)) (CJ, UT s.52): **v**->**v**/**vv**/**v**h
- Vokallängden betecknas med h i ett fåtal ord (kåhl, åhr) (Elk, UT 55): **v**->**v**/ få **v**h
- Aurivillius avvisar att ange lång vokal med h (<ähra, flehra) (A, UT s.62): [ej **v**->**v**h] (från exempelord) [ej **v**r->**v**hr]

1732-1906:

- Swedberg förespråkade inskott av h mellan vokal och konsonant (åhr, ähra, kåhl). Detta att stoppa in ett h mellan vokal och konsonant, finns i 1734 års lag (GP s.163): **vk**->**v**h**k**
- På Salvius tryckeri stryks inskotts-h (ära, år) (GP s.163): [ej **vk** -> **v**h**k**, **v**r->**v**hr]
- I 1743 års lag markeras lång vokal i vissa ord med h, speciellt framför l, r, n och m (åhr, fahran, wahror, muhlbete, mihl, kohl) (UT, s.77
-): **v**l -> **v**hl, **v**r -> **v**hr, **v**m -> **v**hm, **v**n -> **v**hn

aa/å

- **1375-1526**:
- Långt a fick ett bakre rundat uttal vid medeltidens slut. Under 1400-talet skrev vissa skrivare ljudet med å (UT s.9): å -> a/aa/ (ibland) å
- I BU fanns ej aa i form av /å/ (Kl, UT s.34): [ej å -> aa]
- I KrLL var sannolikt det geminerade aa det nya å-ljudet (Kl, UT s.34): å -> aa
- Under 1300-talets senare del förändras kort a-ljud till å-ljud i udd-ljud och framför ng, rd och ld (GP s.148): å- -> a-, ång -> ang, ård -> ard, åld -> ald
- Ibland har kort a via förlängning övergått till å, dels i ordspets (åka, åker, åter). dels framför rdh (gardher ‘gård’, via gärdher). I delar av Sverige uppträder kort a som kort å framför ng, ld, nd, mb och nk (GB s.73): å- -> a-, (årdh -> ardh) ård -> ardh, ång -> ang, åld -> ald, ånd -> and, åmb -> amb, ånk -> ank
- **1526 - 1732:**
- å används för /o/ där /a/ förskjutits till ett långt /o/ (UT s.41): å användes för långt /o/ [ej å->a/aa]
- I 1526 års bibeltryck inkom bokstaven å som uppkom ur det gamla långa a-ljudet (GP s.156) [ej å->a/aa]
- Bokstaven å slog igenom efter att ha tryckts i NT1526 (UT, s.9): [ej å->a/aa]

/k/

1225-1375:

- I YVgL varierar k med c för /k/. /kv/ skrivs qu (quar) (UT 14): k->k/c, kv -> qu
- I CB stavas /k/ med k eller c, geminerat kk. Ibland syns ch eller ck efter n (druncknaþe). /kv/ skrivs qu (CB, UT s.19): k->k/c, k -> (enstaka) ch, nk -> nck, ck->kk, kv -> qu

1375-1526:

- /k/ stavas k, geminerat kk, inte ck. /kv/ skrivs qu (BU, UT 24): ck -> kk, kv -> qu
- /k/ stavas normalt k (skogh, ænkia), men i förbindelse med föregående s ofta sc (wenscap, scogh, scadan) geminerat kk (samtyckio) och ibland ck (samtycke) (Kl, UT s.36): k->k/c/ch, särskilt sk->sk/sc/sch, ck -> kk/(ibland ck)

1526 - 1732:

- I GVB skrivs /k/ normalt k, geminerat systematiskt ck, men även efter l,r och n (folcket). /kt/ skrivs cht. /kv/ betecknas qu (förquaffdhe). I enstaka latinska lånord stavas /k/ med c: scriff, predicadhe (GVB, UT s.43): k -> k/enstaka c/ck, [ej ck -> kk], lk->lck, rk -> rck, nk ->nck, kt -> cht, kv -> qu
- I CJ tecknas /k/ ck efter n, l och r. /k:/ stavas ck och /kt/ skrivs cht (CJ, s.UT 51): lk->lck, rk -> rck, nk ->nck, kt->cht, [ej ck -> kk]
- I Elk skrivs genomgående geminerat /k/ ck och /kt/ stavas systematiskt cht (Elk, UT s.54): kt->cht, [ej ck -> kk
- JS vill behålla cht och q i qw (JS, UT s.70): kt -> cht, kv->qw

1732-1906:

- I 1734 års lag skrivs ck efter l, n och r (folck, marck). /kt/ stavas både cht och kt, gt samt ckt (macht, drächt) (GP s.163): lk->lck, nk->nck, rk->rck, kt -> cht/kt/gt/ckt
- I 1734 lag stavas /kt/ cht (UT, s.78): 1734 lag kt->cht (UT, s.78)
- På Salvius tryckeri skrivs ej cht och ckt, utan istället används kt och gt (163 GP): [ej kt/gt -> cht/ckt]
- År 1801 gav Svenska akademien ut en stavningslära genom Carl Gustaf Leopold, vari /kv/ skrevs qv (GB s.158): kv -> qv
- qu användes för /kv/ långt in på 1800-talet (UT s.9): kv-> qu
- I SAOL 1889, utgåva 6, infördes kv vid sidan av qv (GB s.160). I denna utgåva infördes kv i stället för qv (GP s.166): kv-> kv/qv, [ej kv -> qv] (((motsägelse)))
- Franskt c byttes ut mot k (kaffe, karakter, kapiten, kavaljer, klass, koncert, konselj, kopia, korporal, korrespondens, kredit, kurage, kurtage) (GB s.159). Exempelord har detta k inialt: [ej k- ->vissa c-]
- I utgåvan 1900 av SAOL blev kv obligatoriskt (GB s.160) [ej kv -> qv]

Pal/Affr

1225-1375:

- I YVgL återges palatalisering eller affricering inkonsekvent efter /k/ och /g/ framför æ (kiænnæ, kænnæ) (YVgL, UT s.14) kä ->kiæ (kiæ -> kæ), gä -> giæ (giæ -> gæ)
- I CB tvekan
- (CB, UT s.18)
- I CB är det inkonsekvent återgivning av palatalisering eller affricering vid /k/ och /g/ med i (giærna, gærna, kiöt, skiæg) (CB, UT s.19): kä ->kiæ (kiæ -> kæ), gä -> giæ (giæ -> gæ), kö -> kiö

1375-1526

- I BU syns ingen palatalisering eller affricering vid k och g (kötz, gerningar) (BU, UT 25): [ej kä ->kiæ, kö -> kiö , gä -> gie, gö -> giö]
- 1526 - 1732
- I GVB återges ej palatalisering vid k och g framför främre vokal (källa, kött, göra, gerna (GVB, UT s.43): [ej kä ->kiæ, kö -> kiö, ke -> kie , gö -> giö, ge -> gie]
- Tiällman ville ej ha i efter k och g (gädda, skälla) (NT, UT 66'): [ej kä ->kiæ, kö -> kiö, gä -> giæ]
- JS ville skriva i efter k och g (kiöt, giöda, skiöt) (JS, UT s.70): kö -> kiö, gö -> giö

- Hiärne ville inte utmärka palatalisering av k och g framför främre vokal med i (gängse, köpa, skära) (UH, UT s.74): [ej gä -> giä, kö -> kiö, kä -> kiä]

1732-1906:

- Dahlins har konservativt skriftspråk i Argus och stavar med i mellan k och kort a (betänckia, byggia, kyrkia, sökia), men skriver själv (betänka, bygga, kyrka, söka(GB s.137): -ga -> -gia, -ka -> -kia
- På Salvius tryckeri ströks i eller j mellan g och k innan främre vokal (ske, känna, göra) (GP s.163): [ej ke-> kie, kä -> kiä, gö -> giö]
- I 1734 års lag är affricering av k och g inte konsekvent återgivet (kiöpa, skiäl, kärror, käre, gierning, giödsel, gäst, begiäres) (UT, s.77): kö -> kö/kiö, kä -> kä/kiä, ke -> kie, gö -> giö, ge -> gie, gä -> giä

æ/ä/e

1225-1375

- I YVgL växlar för kort betonad vokal e med æ (feþerne,fæþerne) (YVgL, UT s.13): ä -> e/æ
- I CB växlar e med æ som i YVgL, men æ dominerar (CB, UT s.18): ä -> e/ oftast æ

1375-1526

- I BU växlar e med æ som i YVgL, dock flest e fast æ förekommer (BU, UT s.23): ä -> e/ oftast æ
- I bibeltrycken på 1500-talet används ä istället för æ och ö istället för ø (GP s.156): [ej ä ->æ , ö -> ø]

1526 - 1732

- Stavningsvariationen som tidigare e versus ä återkommer (hierta, fremste, ängel, änkia) (GVB, UT 42): ä -> e/ä
- I GVB växlar ä och e som tecken för kort vokalljud (GP s.160): ä -> e/ä
- I GVB skrivs öppet å-ljud med å och o, särskilt framför ng (lång, long): (GP s.160): å -> o/å
- I CJ används både o och å för kort betonat /o/ (rotta, rost, hopp, ållon, förståndet, spång, månglare) (CJ, UT s.50): å/o-> å/o
- ICJ stavas kort betonat /ɛ/ både med e och ä, även för samma ord (regnar, rägnet, enckia, änckia), jä stavas normalt ie (hielpa) (CJ, UT s.50): e/ä -> e/ä
- Swedberg förordare e och o för de korta vokalljuden (Js, UT s.69): ((()))

1732-1906:

- I 1734 års lag stavas kort betonat /ɛ/ med ä eller e (UT, s.78). ä -> ä/e
  1800-talet
- I SAOL 6, 1889 gäller ä för ä-ljud, särskilt framför r (här pärla) och efter j (järn, hjärta) (GP s.166): [ej är -> er, jä -> je] ((()))

Bilaga 3 Tillåtna ändelser för huvudord och facitord

Tillåtna ändelser för huvudord och facitord

Deklinationer SAOB enl SAG 2:63 —————— SAOB	slutar på	byt föregående ändelse mot		lägg till efter föregående ändelse	
1. gata ——- gater gatn gatur	-a	-or -orna -ona -onar ——- -er -ur	-ors -ornas -onas -onars	-n	-s -ns
2. a) vagn —————— vågn wägn ar wägn er våge n	-**k**			-en -ar -arna -ana -arne -ane -anar -aner -arnar —- -er -ar -n	-s -ens -ars -arnas -anas -arnes -anes -anars -aners -arnars
2.b) by	-**v** (utom a)			-n -ar -arna -ana -arne -ane -anar -aner -arnar —————— -en -er -or -e	-s -ns -ars -arnas -anas -arnes -anes -anars -aners -arnars
2.c) handske	-e	-ar -arna -ana -arne -ane -anar -aner	-ars -arnas -anas -arnes -anes -anars -aners	-n	-s -ns

		-arnar —-------- -a	-arnars		
3.a) balkong balkon	-**k**			-en -er -erna -ena	-s -ens ers ernas enas
3.b) monarki —-------- monarchie	-**v**			-n -er -erna -ena —- -en	-s -ns -ers -ernas -enas
3.c) kafé				-et -n -er -erna -ena —-- -en	-ets -ns -ers -ernas -enas
4.a) ko —---------- kö	-**v** (utom a)			-n -r -rna -na -nar —---- -ren -erna -erne	-s -ns -rs -rnas -nas -nars
5.a) bi —----------	--**v (**utom a)			-et -n -na -ena -erna —---------- -nen -en -erne	-s -ets -ns -nas -enas -ernas
5.b) kvitto				-t -n -na -ena -erna	-ts -ns -nas -enas -ernas
6.a) hus	-**k**			-et -en -ena	-ets -ens -enas

				-erna —-------- -a -es	-ernas
6.b) målar e	-are (kapa ej) —---- kapa e	-na —-------- -ne -arne	-nas -ns	-n —---------	-s -ns
6. c)ordf örand e	-ande/-ende			-n -na -r -rna	-s -ns -nas -rs -rnas
6.d)order —------ ordre ordres	-er	-rar -rarna -rana -rarne -rane -ranar -raner -rarnar	-rar -rarnas -ranas -rarnes -ranes -ranars -raners -rarnars	-n	-s -ns
7.)tricks —- trix trick	-s			-et -en -ena -er -erna	-ets-ens -enas -ers -ernas —---------- -s
tand —-- tänd(er)				-er – ren rena	

Bilaga 4. Återskapade ord; sökta med textord och kopplade till huvudord

Här nedan är en lista med 134 genererade ord som har kunnat sökas upp med textord. De första 104 orden har kunnat hittas i ordboken med metoden exakt matchning, bildade från 119 olika huvudord. De sista 30 orden, är ord som kunnat läggas till och hittats med metoden som tillåter ordböjningar.

Exakt matchning:

tabourette
  TABORETT
  TABURETT
tacksäjelse
  TACKSÄGELSE
tacticus
  TAKTIKER
tactik
  TAKTIK
tafla
  TAFFEL
tagh
  TAG
tahl
  TAL
tahlen
  TALAN
taleman
  TALMAN
tancka
  TANKE
tancke
  TANKE
tappen
  TAPP
tarf
  TARV
tartuf
  TARTUFFE
tecke
  TÄCKE
teriac
  TERIAK
testament
  TESTAMENTE
thara
  TARA
thee
  TE
theologi

TEOLOGI
theologie
TEOLOGI
theorie
TEORI
theriac
TERIAK
thessalonicer
THESSALONIKER
thor
TOR
thora
TORA
thordön
TORDÖN
thornar
TORN
thure
TURE
tidehvarf
TIDEVARV
tidsfördrif
TIDSFÖRDRIV
tidzfördrif
TIDSFÖRDRIV
tienar
TJÄNARE
tienare
TJÄNARE
tienarinna
TJÄNARINNA
tienst
TJÄNST
tierna
TJÄRN
tijd
TID
tijder
TID
tijdh
TID
tijdhende
TIDENDE
tijdhenden
TIDENDE
tijdher
TID
tijender

TIDENDE
TIONDE
tijma
TIMME
tijmar
TIMME
tijond
TIONDE
tijt
TID
tine
TINA
TINNE
tino
TINA
tiog
TJOG
tit
TID
TITTA
tiur
TJUDER
tiänare
TJÄNARE
tiänst
TJÄNS
tiåg
TJOG
tjenare
TJÄNARE
tjenarinna
TJÄNARINNA
togh
TAG
TÅG
toillette
TOALETT
tolamod
TÅLAMOD
tolamodh
TÅLAMOD
tolamoodh
TÅLAMOD
tolck
TOLK
toordön
TORDÖN
torff

TORV
tortt
TÅRTA
tractat
TRAKTAT
traf
TRAV
TRAVE
trafique
TRAFIK
trafvare
TRAVARE
troheet
TROHET
trompeter
TRUMPETARE
troo
TRO
TROD
trool
TROL
tros
TROSS
trotz
TROSS
TROTS
trugh
TRÅG
TRUG
truldom
TROLLDOM
trulldom
TROLLDOM
trwgh
TRUG
trädh
TRÄ
TRÄD
trädha
TRÄDA
träfning
TRÄFFNING
trää
TRÄ
TRÄD
träädh
TRÄD
trääldomb

TRÄLDOM
trään
TREN
tråll
TROLL
tröija
TRÖJA
tufva
TUVA
tungo
TUNGA
tunna
TUNNA
tvedrägt
TVEDRÄKT
twedregt
TVEDRÄKT
twedrächt
TVEDRÄKT
twedrägt
TVEDRÄKT
twedräkt
TVEDRÄKT
twilling
TVILLING
twist
TVIST
twång
TVÅNG
tyckia
TYCKE
tygh
TYG
tysko
TYSKA
täckning
TECKNING
tädh
TÅ
tål
TOLL
töö
TÖ

Ytterligare 30 ord som genererats från 31 huvudord med metoden som tillåter ordböjningar, vilka inte är lika säkra som ovanstående metod (ovanstående ord fångar denna metod också upp).

tanckarne

TANKE
teckelset
TÄCKELSE
tellie
TILJA
testamentz
TESTAMENTE
thebaners
TEBAN
thron
TRO
tijdningar
TIDNING
tijn
TID
tiufwar
TJUGA
tiwffuar
TJUGA
toffta
TOFT
tolamodz
TÅLAMOD
torghet
TORG
treschot
TREDSKA
triumphanter
TRIUMFANT
trodden
TRÅD
troghet
TRÅG
TRÖ
trumpeterna
TRUMPETARE
troldoms
TROLLDOM
tröghet
TRÖ
twagen
TVAGA
twedräckten
TVEDRÄKT
twekan
TVEK
twiflen
TVIVEL

twikan
  TVEK
twingar
  TVING
tyner
  TUNNA
tålken
  TOLK
tällias
  TILJA
tölffter
  TOLFT