Algoritmer för att uttyda ord från svepande gester över virtuella tangentbord

NIKLAS BÄCKSTRÖM
och OSVALD IVARSSON

KTH Datavetenskap
och kommunikation

Examensarbete
Stockholm, Sverige 2012

Algoritmer för att uttyda ord från svepande gester över virtuella tangentbord

N I K L A S B Ä C K S T R Ö M
o c h O S V A L D I V A R S S O N

DD143X, Examensarbete i datalogi om 15 högskolepoäng
vid Programmet för datateknik 300 högskolepoäng
Kungliga Tekniska Högskolan år 2012
Handledare på CSC var Anders Askenfelt
Examinator var Mårten Björkman

URL: www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2012/backstrom_niklas_OCH_ivarsson_osvald_K12010.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Abstract

The small touch screens of smartphones place new demands on text input methods. One popular method is to use different software that allows users to type words by swiping their finger across the screen.

Different implementations of algorithms for translating sweeping gestures into words were compared in this report. These were also compared with the market-leading solution SlideIT. With relatively simple means a combination of algorithms, whose performance is comparable to SlideIT, was successfully implemented.

Sammanfattning

Små pekskärmar hos smartphones ställer nya krav på textinmatningsmetoder. En populär metod är att använda olika program som låter användaren skriva ord genom att svepa fingret över skärmen.

I det här projektet jämfördes olika implementationer av algoritmer för att översätta svepande gester till ord. Dessa jämfördes även med den marknadsledande lösningen SlideIT. Med ganska enkla medel implementerades en kombination av algoritmer vars prestanda är jämförbar med SlideIT.

Fördelning av arbete

Merparten av arbetet utfördes tillsammans. Vi parprogrammerade större delen av systemet och använde versionshantering för att samarbeta om koden. Rapporten skrev vi kollaborativt i Google Docs och vi utbytte idéer och funderingar med varandra under arbetets gång. Båda författarna bidrog lika mycket till arbetet.

Innehållsförteckning

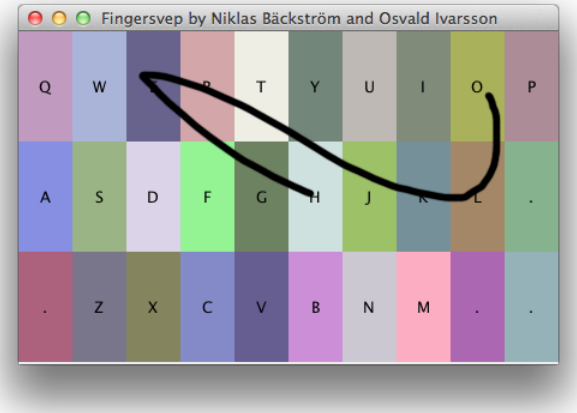
Sida

1 Introduktion 6
2 Problemformulering 7
3 Bakgrund 8
3.1 Färdiga kommersiella lösningar 8
3.2 Algoritmerna bakom gestbaserad textinmatning 8
3.3 Exempel på en färdig algoritm 9
3.4 Maskininlärning 10
4 Metod 11
4.1 Definitioner: tangentbord och svepande gest 11
4.2 Testning 12
4.2.1 Reglerna för straffpoäng 12
4.2.2 Ordlista att använda för ordförslagen 12
4.2.3 Tio testord till utvärdering av algoritmerna 13
4.3 Datainsamling 14
4.4 Definitioner: bokstavsserie och algoritm 15
4.5 Algoritmer 16
4.5.1 Levenshteinavstånd ("Levenshtein Algorithm") 16
4.5.2 Grundläggande algoritm ("Basic Algorithm") 16
4.5.2.1 Striktare variant 16
4.5.3 Förlåtande algoritm ("Forgiving Filtering Algorithm") 17
4.5.4 Ordlängdsalgoritm ("Long Words Algorithm") 17
4.5.5 Fågelvägsalgoritm ("Beelining Algorithm") 18
4.6 Testning av SlideIT 18
5 Resultat 19
5.1 Resultat för bästa algoritmen 19
5.2 Grafer 20
5.2.1 Straffpoäng per algoritm 20
5.2.2 Rätt på första förslaget 21
6 Diskussion 22
6.1 Bästa algoritmen 22
6.1.1 Enkla förbättringar 23
6.1.2 Avancerade förbättringar 23
6.2 Jämförelse med SlideIT 24
6.3 Användbarhet 24
7 Slutsats 25
8 Källförteckning 26
Appendix A: Testningsresultat 27
Appendix B: Urval av testningsutskrifter 28

1 Introduktion

Smartphones blir allt vanligare nu för tiden. En vanlig egenskap hos denna typ av telefon är att den saknar ett fysiskt tangentbord, vilket gör att användaren istället får skriva på något slags virtuellt tangentbord. Den traditionella typen av virtuella tangentbord härmar mekaniska tangentbord - man trycker och släpper fingret på en viss bokstav för att skriva den. Detta kräver en viss precision av användaren. Om man inte trycker på precis den bokstaven man vill skriva så blir det fel och detta gäller för varenda bokstav. Dessutom är många smartphone-skärmar ganska små vilket gör det ännu svårare att trycka på rätt ställe.

Det finns dock alternativa inmatningsmetoder till att trycka på en bokstav i taget. En populär metod är att använda ett virtuellt tangentbord som bygger på svepande gester. Sådana har implementerats för smartphones med god framgång. Istället för att trycka precist upprepade gånger sätter man bara ned fingret på första bokstaven i ordet man vill skriva och sveper sedan förbi alla bokstäver i ordningen de ska förekomma. När fingret sedan lyfter (vid sista bokstaven i ordet) ser man på banan fingret rörde sig längs. Detta utgör den svepande gesten. En bakomliggande algoritm analyserar sedan gesten för att uttyda vilket ord användaren menar att skriva. Nedan följer ett exempel på hur en svepande gest kan se ut.



Figur 1

Exempel på svepande gest för ordet "HELLO". Användaren börjar genom att klicka och dra vid H, fortsätter upp till E, gör en skarp sväng och drar sedan vidare i en svepande rörelse till L för att till sist avsluta gesten vid O.

2 Problemformulering

Det finns många frågeställningar kring användandet av svepande gester för inmatning av text. Syftet med det här projektet var endast att implementera och undersöka olika algoritmer som skulle kunna vara lämpliga att använda med den ovan beskrivna textinmatningsmetoden.

En viktig deluppgift var att kunna ge flera ordalternativ till användaren när resultatet var tvetydigt. I denna uppgift ingick också att rangordna dessa alternativ enligt hur troligt det var att respektive ord var just det som söktes.

Det här projektet fokuserade endast på de tekniska delarna av problemområdet. Frågor inom området människa-datorinteraktion, som till exempel om inmatningsmetoden är användbar eller effektiv, låg utanför projektets omfång.

3 Bakgrund

3.1 Färdiga kommersiella lösningar

Det finns flera företag som har kommersiella produkter vilka kan göra om svepande gester på pekskärmar till text. Ett exempel är företaget Dasur Pattern Recognition Ltd.[1] som säljer en app, till bland annat Androidplattformen, vilken möjliggör just gestbaserad textinmatning. Deras app, SlideIT[2], säljs för 39,88 SEK och har passerat 500 000 köp på Google Play, vilket visar att det är ett populärt sätt att skriva på.

Ett annat företag är Swype[3] som har en liknande lösning. Den går dock i nuläget (2012-03-31) inte att köpa som en fristående app till Androidplattformen utan man måste köpa en smartphone som har Swype förinstallerat.

3.2 Algoritmerna bakom gestbaserad textinmatning

Det ligger många års forskning och flera patentansökningar bakom tekniken till både Swype och SlideIT. Detta gör att det inte finns så mycket offentlig information om hur själva algoritmerna fungerar (bortsett från patentansökningarna[4]).

Extremetech[5] har dock satt sig in i Swypes patent och har i stora drag förklarat hur Swype fungerar. Swype bygger på flera olika algoritmer och lösningar. Enligt Extremetech har Swype en lista med tänkbara ord som användaren kan tänkas vilja skriva. Sedan gör Swype om varje svepande gest till en modell som sparar vilken bana gesten sveptes över. Modellen sparar vilka bokstäver (senare benämnt som en bokstavsserie) som gesten gick över och sedan jämförs modellen med möjliga banor för de olika orden i ordlistan. Efter detta presenteras användaren med olika ord som hon får välja mellan.

[1] http://www.mobiletextinput.com/
[2] https://play.google.com/store/apps/details?id=com.dasur.slideit
[3] http://www.swype.com/
[4] http://www.freepatentsonline.com/7453439.html
[5] http://www.extremetech.com/extreme/97837-how-does-swype-really-work

Swype använder även flera olika heuristiker[6] för att förbättra lösningsförslagen. Några heuristiker är följande:

1. Delar av gesten som sveps snabbare anses vara mindre precisa än delar som sveps långsamt.
2. Skarpa svängar i gesten gör det mer troligt att bokstaven under gesten ska tas med, istället för att bara vara en utfyllnadsbokstav på vägen till de riktiga bokstäverna.
3. Ord i ordlistan, som omöjligen kan vara kandidaer, filtreras bort. Ett ord i ordlistan anses vara omöjligt om inte alla dess bokstäver ligger på, eller nära, gestens bana.
4. Vanliga ord prioriteras högre än ovanliga ord.

Utöver detta försöker Swype hela tiden att lära sig vilka ord användaren skriver och hur användaren skriver dessa. Detta gör att Swype hela tiden förbättrar sin förmåga att ge korrekta ordförslag.

3.3 Exempel på en färdig algoritm

Utöver det man kan få fram ur Swypes patentansökan finns det inte särskilt mycket information om hur algoritmerna faktiskt fungerar. Det finns dock en algoritm[7], beskriven i en artikel på Internet skriven av Krishna Bharadwaj, som översätter bokstavsserier till tänkbara ord. Algoritmen är skriven i programmeringsspråket Python och är endast 61 rader lång. Den kräver, precis som Swype och SlideIT, tillgång till en ordlista med tänkbara ord. Den förutsätter också att den svepande gesten redan är översatt till en bokstavsserie. Sedan använder algoritmen följande tre heuristiker:

1. Den filtrerar fram alla ord som börjar och slutar på samma bokstav som den svepande gestens bokstavsserie börjar och slutar på.
2. Den filtrerar fram alla ord vars bokstäver förekommer i rätt ordning i den svepande gestens bokstavsserie (Swype gör samma sak, men inte lika strikt).
3. Den räknar ut en gissning på det sökta ordets minimumlängd baserat på antalet hopp mellan olika rader på tangentbordet. Alla ord som understiger den sökta minimumlängden filtreras slutligen bort.

Krishnas algoritm är, jämfört med Swype och SlideIT:s algoritmer, ganska enkel, och skulle kunna vara en utgångspunkt för det här projektet.

[6] Heuristik: Ett sätt att göra kvalificerade gissningar till problem som är för svåra att lösas exakt.
[7] http://krishnabharadwaj.info/how-swype-works/

3.4 Maskininlärning

Ett annat sätt att angripa problemet med att tyda gester är att använda metoder baserade på maskininlärning. Taligenkänning samt andra problem inom signalanalys kan lösas med dessa metoder vilket indikeras i Hidden Markov Models (Rabiner, 1989). I publikationen beskrivs det hur maskininlärning kan uppnås med hjälp av dolda Markovmodeller. Det är möjligt att Swype, med flera, använder denna teknik för att åstadkomma sin lärande funktion.

På en väldigt generell nivå kan man säga att dolda Markovmodeller är ett matematiskt system som till exempel kan användas till att känna igen mönster. Det kan även tränas för att bli bättre och bättre på detta. Problemet i det här projektet, att känna igen vilket ord användaren försöker svepa fram, skulle kunna lösas med denna teknik.

4 Metod

Algoritmerna är tvungna att, när de analyserar en gest, vara tillräckligt intelligenta för att förstå vilket ord användaren vill skriva. Svepande gester passerar, förutom de bokstäver som är tänkta att ingå i ordet, oftast en del oönskade bokstäver. Dessa bokstäver utgör ett brus i gesten. Algoritmerna måste därför utföra någon slags signalanalys för att hantera detta. För att exempelvis skriva ordet "TOP" passerar man typiskt bokstäverna "**T**YUI**OP**" med en svepande gest. Bokstäverna "YUI" utgör då bruset i gesten (observera hur bokstäverna "**T**YUI**OP**" ligger bredvid varandra på ett vanligt svenskt tangentbord).

Tvetydighet är också ett problem. För att exempelvis skriva det engelska ordet "POOL" passerar man typiskt bokstäverna "**POL**" med en svepande gest. I detta fall finns inga brusbokstäver med i gesten, men ett annat tänkbart ord skulle kunna vara "POLL". Rätt kandidater måste föreslås och sedan rangordnas på ett önskvärt sätt. Det finns många olika tänkbara sätt att göra denna klassificering på och i det här projektet utvärderas olika metoder i sökandet efter en effektiv och beräkningsmässigt rimlig lösning.

4.1 Definitioner: tangentbord och svepande gest

Ett tangentbord K definieras som en tvådimensionell matris av bokstäver. Varje bokstav omsluts av en rektangel med bredd K_w och höjd K_h och alla dessa rektanglar är lika stora. Bokstäver i det engelska alfabetet placeras ut enligt QWERTY-arrangemanget[8] på tangentbordet. Detta ger upphov till den totala storleken $3 * 10$ (varav fyra rutor förblir tomma och representeras med punkter). Tangentbordet är således $10 * K_w$ pixlar brett och $3 * K_h$ pixlar högt. Ett koordinatsystem över tangentbordet, vars origo ligger i det övre vänstra hörnet, definieras också. Koordinatsystemets x-axel är positiv åt höger och y-axeln är positiv nedåt.

En svepande gest G definieras som en serie av stickprov $G_1, G_2, ..., G_n$. Varje stickprov G_i består av den tvådimensionella punkt $(G_i[x], G_i[y])$ på tangentbordet vars heltalskoordinater motsvarar var pekaren befinner sig för tillfället. Ett stickprov tas i varje punkt som pekaren passerar över tangentbordet, men endast när pekaren är nedtryckt (för att kunna markera start- och slutpunkt för gesten). Stickprov där $x \notin [0, 10 * K_w]$ eller $y \notin [0, 3 * K_h]$ förkastas.

[8] http://en.wikipedia.org/wiki/Qwerty

4.2 Testning

Algoritmerna som utvecklades i det här projektet behövde kunna utvärderas, så en testningsmetod skapades för att ge underlag till denna utvärdering. Testningsmetoden använde en uppsättning sparade gester för att testa algoritmerna. Varje sparad gest lagrades tillsammans med det ord den var tänkt att motsvara. Testningen gick till så att algoritmerna fick ge ordförslag för varje sparad gest. Det kunde då verifieras om algoritmerna föreslog rätt ordalternativ. Genom att dela ut straffpoäng för felaktiga svar skapades en uppfattning om algoritmernas prestanda. Till exempel delades straffpoäng ut när en algoritm föreslog ord som inte eftersöktes. Ytterligare straffpoäng delades ut om algoritmen inte lyckades föreslå det sökta ordet över huvud taget. För att få en övergripande uppfattning om en algoritms prestanda testades den på samtliga sparade gester varefter straffpoängen summerades. En låg straffpoängsumma indikerade att algoritmen presterade bra, medan en hög straffpoängsumma indikerade motsatsen. Med den här testningsmetoden kunde uppdateringar till algoritmerna hela tiden utvärderas under projektets gång.

4.2.1 Reglerna för straffpoäng

Varje algoritm började på 0 poäng. För varje svept gest och motsvarande ordförslag utvärderades sedan förslagen och eventuella straffpoäng adderades till totalsumman. Reglerna för straffpoäng var följande:

1. Om algoritmen föreslog rätt ord som första alternativ tilldelades inga straffpoäng.
2. Om algoritmen föreslog rätt ord, men *inte* som första alternativ, tilldelades 1 straffpoäng för varje felaktigt förslag innan det rätta ordet.
3. Om algoritmen inte alls föreslog rätt ord tilldelades antingen 4 straffpoäng, eller 1 poäng för varje felaktigt förslag, beroende på vilket värde som var *störst*.

4.2.2 Ordlista att använda för ordförslagen

Engelskan är ett betydligt större språk än svenskan, vilket gjorde att det var enklare att hitta engelska ordlistor än svenska. Därför fokuserade det här projektet på engelska ord och en engelsk ordlista[9] anskaffades. Skulle det efter projektets slut vara önskvärt att byta språk till svenska är det en trivial sak att istället ladda in en svensk ordlista och sedan lägga till "ääö" till det virtuella tangentbordet. Detta går självklart lika bra att göra för många andra språk också. För att uppnå önskvärd användbarhet, med det svenska språket inlagt, kan det dock vara nödvändigt att bygga ett mer avancerat stöd för sammansatta ord. Detta beror på att denna typ av ord förekommer oftare, och är viktigare, i just svenskan.

[9] Engelsk ordlista (nedladdad från http://dl.dropbox.com/u/17629670/serve/wordlist.txt) med totalt 41 208 ord.

4.2.3 Tio testord till utvärdering av algoritmerna

Totalt valdes tio testord ut för utvärdering, med hänsyn till att alla orden skulle vara olika placerade på tangentbordet så att gesterna skulle se olika ut. Orden matades in med svepande gester som sedan sparades. Algoritmernas prestanda optimerades sedan kring resultaten för dessa testord. Det är dock viktigt att lägga märke till att det var projektets målsättning att algoritmerna även skulle kunna tyda gester för i stort sett vilka andra ord i ordlistan som helst. Se Tabell 1 för de tio testorden samt tillhörande motiveringar om varför de är intressanta som testord.

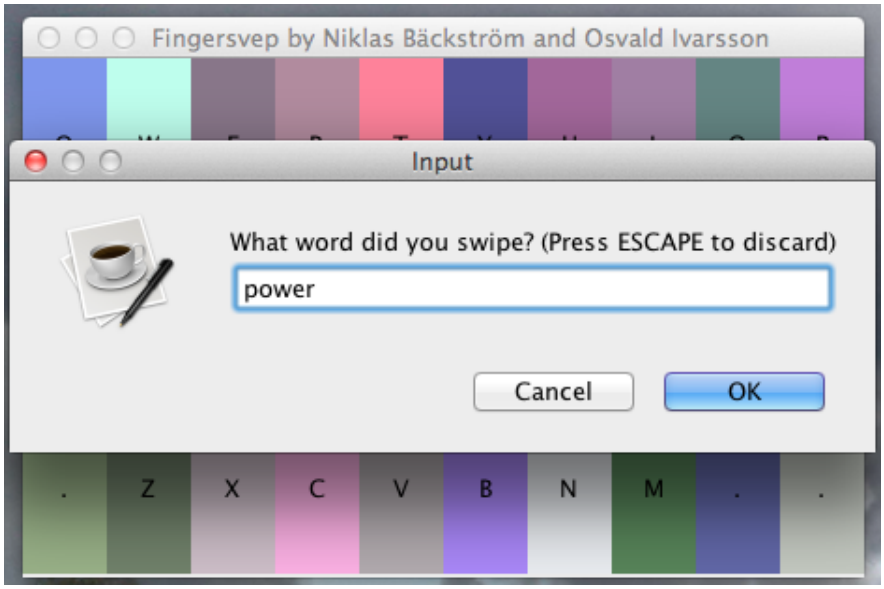
absinthe	Detta är ett ganska långt ord. Gesten innehåller typiskt många skarpa svängar. Det bör dock vara lätt att uttyda på grund av att det finns få liknande ord, och således bör gesten för ordet vara förhållandevis unik.
power	Gesten för detta ord täcker typiskt endast en rad och en sväng fram och tillbaka.
query	Gesten för detta ord täcker också typiskt endast en rad och innehåller flera svängar fram och tillbaka.
winning	Gesten för detta ord innehåller flera svängar fram och tillbaka på samma del av tangentbordet.
lag	Gesten för detta ord täcker endast en rad. Ordet är väldigt kort men har en förhållandevis lång gest.
add	Detta ord är väldigt kort och även gesten är kort. Det blir också osäkert om ordet "add" eller "ad" ska föreslås eftersom deras gester ser likadana ut.
coral	Gesten för detta ord är förhållandevis lång.
attack	När detta ord sveps fram bör det vara ganska lätt att missa bokstaven C eller att bara vidröra den knappt.
pool	Gesten för detta ord är väldigt kort. Med samma resonemang som för "add" är det osäkert om "pool" eller "poll" borde föreslås.
burnt	Gesten för detta ord kan efterlikna en cirkel men användaren kan också dra raka streck mellan bokstäverna och åstadkomma en något kortare gest.

Tabell 1
De tio utvalda testorden.

4.3 Datainsamling

För att få tillräckligt mycket data fick fyra personer skriva in de tio testorden med svepande gester minst tio gånger per ord. Förhoppningen var att gesten för ett visst ord skulle se lite annorlunda ut varje gång den sveptes fram, speciellt om man jämförde gester för ett visst ord från olika personer. Resultatet blev gester som var någorlunda olika men som ändå motsvarade samma ord. Algoritmerna förväntades förstås kunna klara av detta.

För att kunna genomföra insamling av dessa gester implementerades ett virtuellt tangentbord och den tillhörande inmatningsmetoden som läste in gesterna. Java valdes som utvecklingsmiljö för projektet. Det grundläggande program som behövde distribueras till testpersonerna sattes sedan upp. Den tidiga versionen kunde inte uttyda några ord, eftersom den bara användes till just datainsamling. Efter att tangentbordet hade färdigställts och datainsamlingen hade utförts skedde utvecklingen av själva algoritmerna, som skulle uttyda ord från gesterna och rangordna dem. Själva programmet kallades för "Fingersvep".



Figur 2
Programmets insamlingsläge som låter användaren svepa en gest och sedan ange vilket ord hon menar att gesten motsvarar.

4.4 Definitioner: bokstavsserie och algoritm

En bokstavsserie S för en gest G definieras som den serie av bokstäver som gestens bana passerar på tangentbordet. Bokstavsserien är då resultatet av någon beräkning $S = f(G)$. Det finns olika typer av bokstavsserier och dessa går att räkna ut på olika sätt.

Oviktade bokstavsserier definieras som S_{ULS} ("Unweighted Letter Series") där $ULS(G)$ är den funktion som för varje stickprov G_i ser om $(G_i[x], G_i[y])$ täcker någon av bokstävernas rektanglar. Om så är fallet, och bokstaven inte är den senast tillagda bokstaven i serien, så läggs bokstaven till. S_{ULS} innehåller alltså, i ordning, alla bokstäver vars rektangulära yta passeras av pekaren. Samma bokstav kan förstås förekomma flera gånger i serien men om två på varandra följande stickprov täcker samma bokstav så tas alltså bokstaven bara med en gång i den delen av serien.

Viktade bokstavsserier definieras som S_{WLS} ("Weighted Letter Series"), där $WLS(G)$ är den funktion som för varje stickprov G_i mäter avståndet från stickprovets punkt $(G_i[x], G_i[y])$ till samtliga näraliggande bokstäver på tangentbordet. Bokstaven som är närmast och nära nog (inom någon viss gräns) läggs till i serien. Avståndet till denna bokstav sparas också så att man ser precis hur nära pekaren passerade bokstaven, och därmed hur troligt det är att den ingår i ordet som eftersöks.

En algoritm definieras som en funktion A av en graf G. Funktionen ger upphov till ett algoritmresultat R bestående av en lista av kandidater. En kandidat är ett ord från ordlistan som algoritmen anser att användaren skulle kunna ha velat skriva med sin svepande gest. En filtrerande algoritm är en algoritm som för varje ord w, i ordlistan, evaluerar ett antal filter. Varje filter svarar antingen ja, om det anser att w ska vara en kandidat, eller nej, om w ska förkastas. w anses endast vara en kandidat om samtliga filter svarar ja.

4.5 Algoritmer

Ett flertal olika algoritmer utvecklades för att lösa problemet med att uttyda ord från svepande gester. Vissa av algoritmerna är helt fristående medan andra är påbyggnader av tidigare algoritmer för att finjustera prestandan. De flesta algoritmerna designades av oss själva men några härstammade också från litteratur och andra resurser, i sin helhet eller till någon grad.

4.5.1 Levenshteinavstånd ("Levenshtein Algorithm")

Projektets första algoritm baseras helt och hållet på beräkningar av s.k. Levenshteinavstånd[10] (på engelska ofta känt som "edit distance"). Sådana avstånd anger, med ett heltal, hur stor skillnad det är mellan två strängar av bokstäver. Algoritmen för att beräkna Levenshteinavstånd används i projektet för att beräkna avståndet från $ULS(G)$ till varje ord i ordlistan. I princip mäts antalet bokstäver som behöver tas bort eller sättas in för att serien ska bli ekvivalent med ordet. Det ord med lägst avstånd till bokstavsserien räknas som den mest sannolika kandidaten, och ett godtyckligt antal kandidater kan väljas på det här sättet.

4.5.2 Grundläggande algoritm ("Basic Algorithm")

Delar av Krishna Bharadwajs algoritm, som nämndes tidigare, implementeras för att få en grundläggande algoritm. De två första av hans tre föreslagna filter används och uppskattas ge ett bra grundurval av kandidater. I den grundläggande algoritmen anses ett ord, i ordlistan, vara en kandidat precis när följande två kriterier uppfylls:

1. $ULS(G)$ innehåller samtliga av ordets bokstäver i ordning.
2. Ordets första och sista bokstav står i $ULS(G)$ första respektive sista position.

4.5.2.1 Striktare variant

Betrakta nu fallet när användaren sveper gesten för "RINGING", vilket ger upphov till $ULS(G) =$ "**R**TYU**I**KJ**N**BH**G**HJU**I**KJ**N**BH**G**" eller något liknande (brusbokstäverna kan variera). Den grundläggande algoritmen accepterar "RINGING" som en kandidat, men även "RING". Varför gör den det? För både "RINGING" och "RING" gäller det att samtliga bokstäver i ordning ingår i $ULS(G)$ samt att första och sista bokstaven står i seriens första respektive sista position. Samtliga bokstäver i "RING" uppträder dock i ordning långt innan serien tagit slut. Om användaren hade velat skriva "RING" hade det antagligen inte uppträtt ytterligare tio skräptecken efter att sista bokstaven i ordet passerats. Alltså bör ett ord bara anses vara en kandidat när inget mindre än samtliga bokstäver i serien måste gås igenom för att hitta ordets bokstäver i ordning. Krishnas andra filter utvidgas med detta tillägg.

[10] http://en.wikipedia.org/wiki/Levenshtein_distance

4.5.3 Förlåtande algoritm ("Forgiving Filtering Algorithm")

Den förlåtande algoritmen implementeras som en påbyggnad av den filtrerande algoritmen. Denna algoritm ska kunna hantera fallet då användaren råkar avsluta sin svepande gest för tidigt eller för sent. Låt säga att användaren sveper gesten för "FAD" och att det resulterar i $ULS(G) =$"**F**DS**A**S". Observera att sista bokstaven i ordet saknas i slutet av serien. "FAD" kommer alltså på grund av detta fel inte att föreslås som kandidat av till exempel den grundläggande algoritmen. Den förlåtande algoritmen, i fallet då inga kandidater hittas, prövar att för varje bokstav runt "S" (sista bokstaven i serien) evaluera en gest G', vilket är gesten G med den extra bokstaven tillagd. Med $ULS(G') =$ "**F**DS**A**S**D**", vilket motsvarar fallet när algoritmen prövar att lägga till "D", föreslås "FAD" som kandidat. På samma sätt kan gesten ta slut för sent vilket resulterar i skräptecken i slutet av serien, exempelvis $ULS(G) =$ "**F**DS**A**S**D**F". Den förlåtande algoritmen prövar att ta bort bokstäver från slutet av serien för att se om några kandidater kan föreslås.

4.5.4 Ordlängdsalgoritm ("Long Words Algorithm")

Denna algoritm implementeras som en påbyggnad av den grundläggande algoritmen. Den baseras på en egen idé om att ju *längre* en gest är desto troligare är det att användaren vill skriva ett långt ord. Algoritmen utnyttjar precis detta och sorterar alla föreslagna kandidater i omvänd ordning enligt deras längd, så att långa kandidatord föreslås före korta.

4.5.5 Fågelvägsalgoritm ("Beelining Algorithm")

De ovan nämnda algoritmerna har, trots sin strikthet, fortfarande möjlighet att föreslå många inkorrekta kandidater. Låt säga att användaren sveper gesten för "POWER" vilket ger $ULS(G) =$"**PO**IUYTRE**WER**". "POWER" föreslås som kandidat men även "POTTER": "**PO**IUY**T**R**E**WE**R**". Observera att det då finns två skräptecken mellan "E" och "R", men att dessa två bokstäver ligger precis bredvid varandra på tangentbordet! Antagandet görs att användaren inte tar väldigt onödiga omvägar mellan par av bokstäver i ordet. Ett filter konstrueras som avvisar ord som skulle ha accepterats med just denna typ av omvägar. Detta filter utgör ett tillägg till den grundläggande algoritmen.

Algoritmen studerar de bokstäver i serien som valts ut för att ingå i ett kandidatord. Funktionen $D(a,b)$ anger det högsta antalet tillåtna skräptecken som får uppträda i serien mellan dessa två utvalda bokstäver. Exempelvis gäller att $D("E","R") = D("R","E") = 0$ då dessa två bokstäver ligger bredvid varandra på tangentbordet. Generellt grundas värdet av $D(a,b)$ på Manhattanavståndet[11] mellan a och b på tangentbordet. Värdet sätts sedan till en längdenhet mindre än Manhattanavståndet. Detta därför att om avståndet mellan två bokstäver exempelvis är två längdenheter så finns det en skräpbokstav där emellan, och värdet för funktionen bör då således vara en längdenhet. För att tillåta omvägar, till en viss grad, adderas två längdenheter till värdet om värdet redan överstiger noll längdenheter. Alltså är algoritmen något förlåtande, såvida inte bokstäverna ligger bredvid varandra på tangentbordet, i vilket fall användaren måste svepa raka vägen mellan dessa bokstäver. Annars räcker det med att svepa ett streck som är någorlunda rakt.

4.6 Testning av SlideIT

För att kunna jämföra projektets resultat med en befintlig kommersiell lösning testas det virtuella tangentbordet SlideIT. En användare får skriva alla tio testord, tio gånger per ord, på en vanlig smartphone med detta tangentbord. Ordförslagen skrivs sedan ned och tilldelas straffpoäng på samma sätt som för de övriga algoritmerna.

[11] http://en.wikipedia.org/wiki/Manhattan_distance

5 Resultat

Den insamlade testdatan, bestående av 400 gester, användes med den tidigare föreslagna testningsmetoden för att evaluera olika kombinationer av algoritmer. Resultaten presenteras i stapeldiagram i Figur 3 och Figur 4. De exakta siffrorna presenteras i Appendix A. Levenshteinavståndsalgoritmen, den grundläggande algoritmen och fågelvägsalgoritmen testades för sig själva samt med olika möjliga kombinationer av den förlåtande algoritmen och ordlängdsalgoritmen. Den grundläggande algoritmen testades också i sin mindre strikta variant.

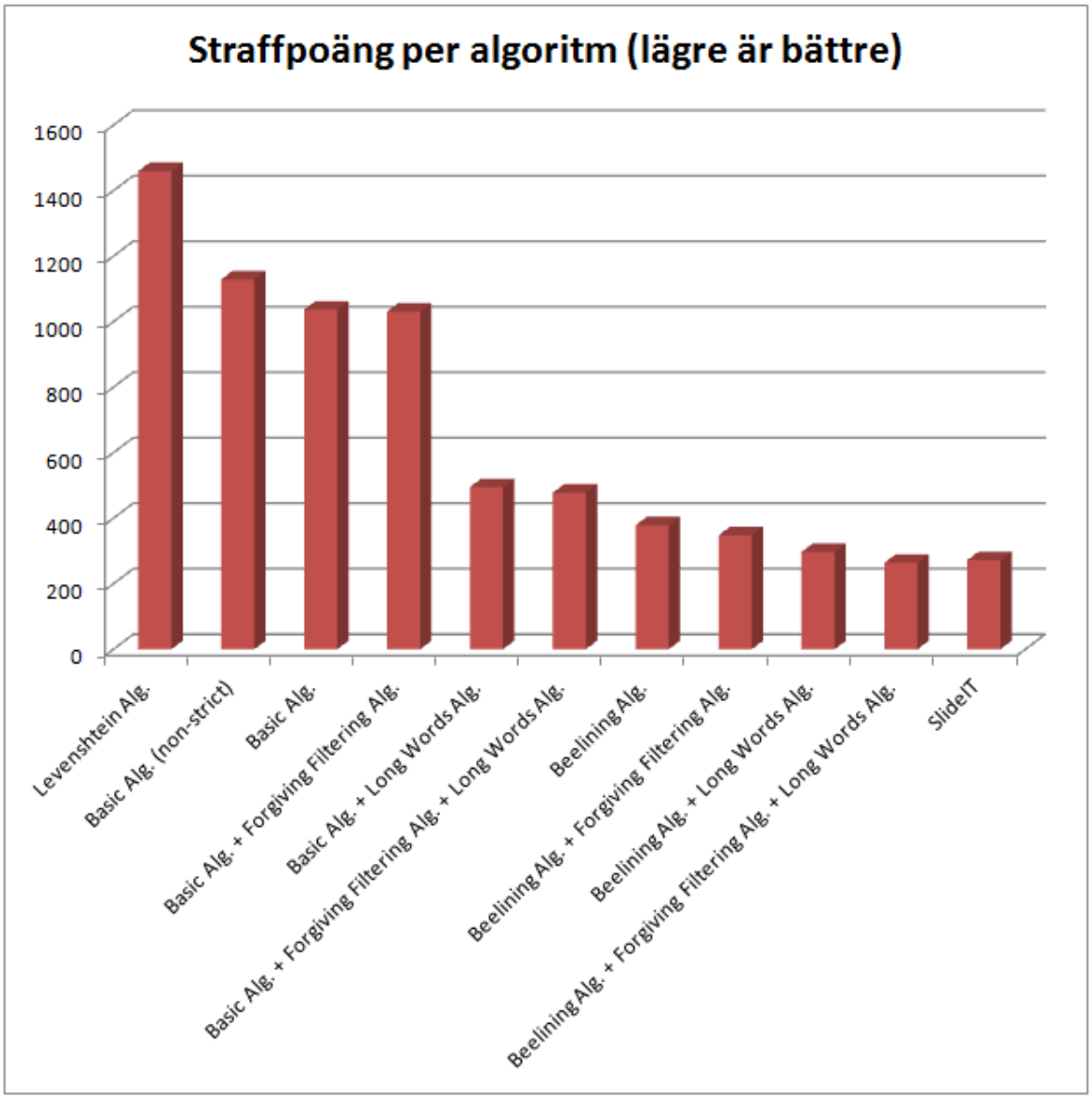
Testningen av SlideIT resulterade i att dess straffpoäng blev totalt 68 och att antalet korrekta gissningar på första försöket var 70 (av 100). Detta resultat skalades upp för att kunna visas i samma storleksordning som resten av resultaten.

5.1 Resultat för bästa algoritmen

Den bästa algoritmen var en kombination av fågelvägsalgoritmen (vilken utökar den grundläggande algoritmen), den förlåtande algoritmen samt ordlängdsalgoritmen. När denna kombination av algoritmer utvärderade alla 400 sparade gester från testpersonerna lyckades den föreslå 298 av 400 ord korrekt på första försöket. Straffpoängen blev 264. Majoriteten av straffpoängen delades ut på grund av att ord som "CHORAL", "COITAL", "CHORDAL" och "CORRAL" föreslogs före ordet "CORAL" samt att ordet "POLL" föreslogs före ordet "POOL". Resten av straffpoängen orsakades mest av att vissa gester var speciellt otydliga kombinerat med att algoritmerna var alltför oförlåtande i dessa fall. Evalueringstiden var inte märkbar och således gjordes ingen exaktare mätning.

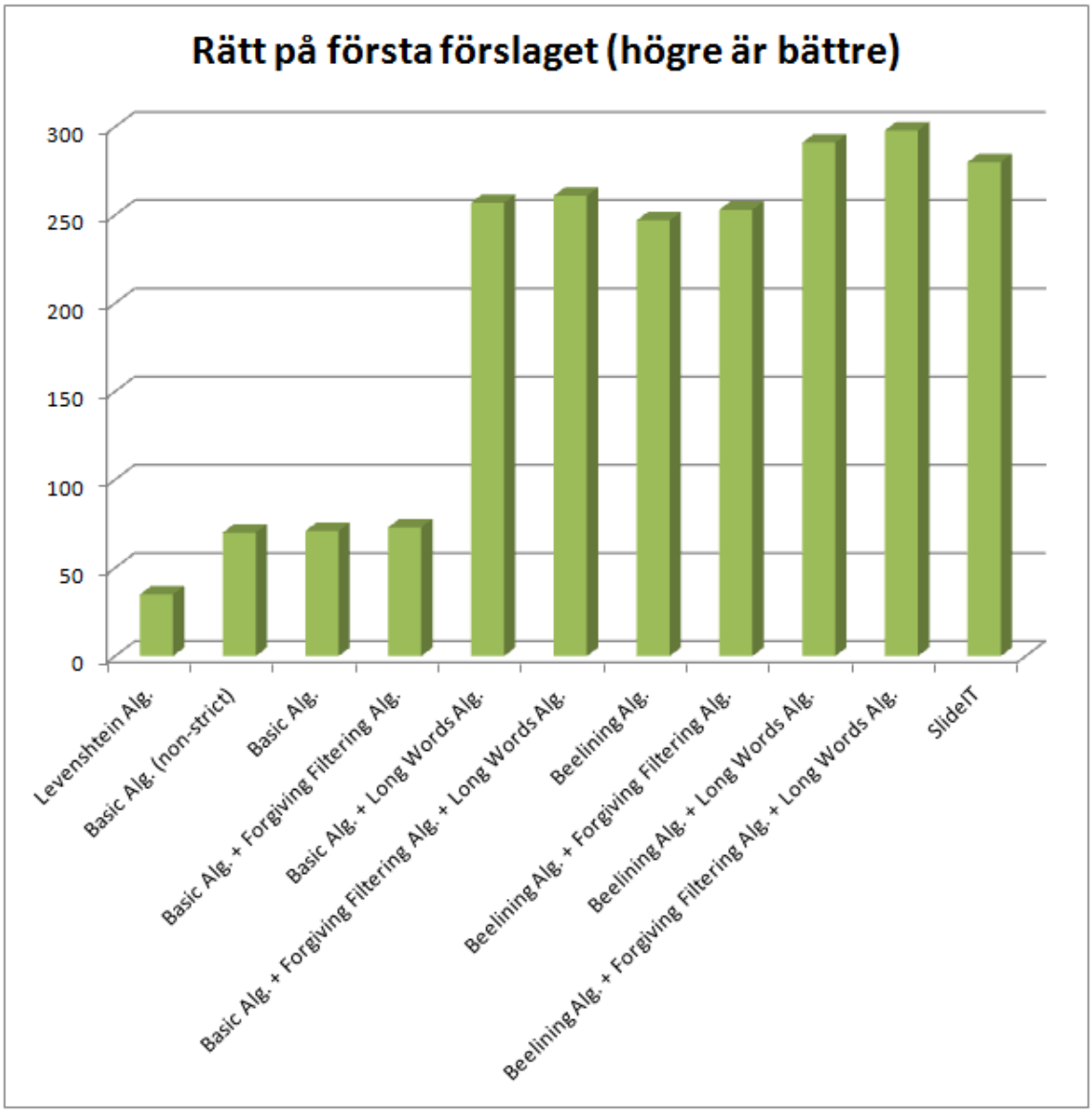
5.2 Grafer

5.2.1 Straffpoäng per algoritm



Figur 3

5.2.2 Rätt på första förslaget



Figur 4

Se Appendix B för en listning av testningsutskrifter för den grundläggande algoritmen (av den icke-strikta varianten) samt den kombination av algoritmer som presterade bäst i testningen.

6 Diskussion

För ett väldigt grundläggande program som översätter svepande gester till ord behöver man inte forska i tio år eller implementera patenterade maskininlärningsalgoritmer, vilket har visats i det här projektet. Som man kan se i resultaten får den bästa algoritmen i det här projektet en hög träffsäkerhet (298 av 400 svepta ord på första förslaget) med relativt enkla medel. Ingen av algoritmerna i projektet bygger på maskininlärning eller anpassar sig efter användaren, vilket de marknadsledande produkterna däremot gör. Trots detta presterade algoritmerna väldigt bra, tack vare optimeringarna som utfördes tillsammans med försöken att hela tiden reducera antalet kandidater.

6.1 Bästa algoritmen

Som man ser i resultaten är den bästa algoritmen i själva verket en kombination av flera algoritmer. Kombinationen är relativt välpresterande jämfört med SlideIT då den både får lägre straffpoäng och oftare föreslår rätt kandidat på första försöket.

Vidare ser man att resultaten förbättrades mycket av att långa kandidater valdes före korta. Denna sortering, som görs av ordlängdsalgoritmen, utgör i själva verket hela rangordningsdelen av lösningen. Sorteringen är dessutom implementeringsmässigt väldigt enkel. De andra algoritmerna är baserade på filtrering och utför därför ingen rangordning. Resultaten skulle kunna förbättras ytterligare om alla algoritmer var baserade på rangordning istället för filtrering. Om ett filter svarar nej för en kandidat skulle algoritmerna, istället för att helt utesluta kandidaten, kunna tilldela den en lägre sannolikhet.

Att evalueringen genomfördes snabbt, trots att samtliga ord i ordlistan behandlas för varje gest, berodde till stor del på att ordlistan som användes var ganska kort. Om en större ordlista anskaffades skulle evalueringen gå betydligt långsammare. Det kan dock avhjälpas med indexering[12]. Stora ordlistor medför också problemet att det krävs mycket lagringsplats, vilket kan avhjälpas med komprimering.

För att göra det riktigt användbart för användaren och förbättra ordförslagen ännu mer krävs dock fler förbättringar. Nedan anges några förslag som skulle kunna förbättra algoritmernas prestanda.

[12] Index: Datastruktur vars uppgift är att förkorta tiden det tar att läsa data från en datamängd.

6.1.1 Enkla förbättringar

Swype, med flera, föreslår *vanligt förekommande* ord före ovanliga ord. Det första steget i att åstadkomma detta är att anskaffa en viktad ordlista med de mest (och minst) använda orden i till exempel det engelska språket. Om man sedan sorterar kandidaterna baserat på dessa vikter kommer de vanligaste orden att föreslås först, vilket bör leda till bättre förslag överlag för majoriteten av användarna. Som utbyggnad av det här kan man även öka vikten varje gång användaren väljer ett ord. Detta gör att systemet allt eftersom "lär sig" vilka ord användaren använder mest och föreslår dessa oftare. En algoritm som gör allt detta skulle vara en bra ersättning till ordlängdsalgoritmen.

Det andra förslaget bygger också på att låta systemet lära sig mer och mer om användaren. En enkel förbättring är att låta användaren skriva in ord som hon vill ska finnas i ordlistan. Systemet ger sedan dessa ord ganska höga vikter. Efter det hanterar man dessa ord precis som de ord som redan ligger i ordlistan. Utan den här förbättringen skulle användaren antagligen bli väldigt frustrerad av att vissa ord inte finns i ordlistan och aldrig heller föreslås.

Det tredje förslaget är att använda Krishnas tredje filter, vilket diskuterades tidigare. Filtret räknar ut en uppskattad minimumlängd på ordförslagen, vilket gör att antalet ordförslag antagligen kan minskas ännu mer. Filtret påminner en del om ordlängdsalgoritmen. Det filtrerar dock bort ord som understiger minimumlängden helt och hållet, medan ordlängdsalgoritmen bara ger kortare ordförslag lägre prioritet.

Det sista förslaget är att använda en slags tvåvägsalgoritm. Den är en variant av Krishnas andra filter som söker fram ordets bokstäver i bokstavsserien från båda hållen samtidigt. Om sökandet möts i mitten och en bokstav saknas precis där, för att den missades av gesten, så kan den läggas till och ordet föreslås som kandidat ändå. Detta tillåter alltså att användaren missar någon enstaka bokstav i mitten av gesten.

6.1.2 Avancerade förbättringar

Den största förbättringen annars, som dock skulle innebära en total omskrivning av algoritmerna, är att använda maskininlärning och implementera en lösning med hjälp av dolda Markovmodeller. Då skulle man kunna sätta upp en matematisk modell där man inte jobbar med rena bokstavsserier utan istället använder de punkter som användaren sveper över på tangentbordet. Det skulle innebära att man går över till mycket mer avancerade algoritmer som är svårare att sätta sig in i. Maskinlärning skulle åstadkommas genom att uppdatera de dolda Markovmodellerna allt eftersom nya gester sveps och systemet ser vilket ord användaren väljer från listan av kandidater. Detta skulle leda till att systemet förbättrar sig baserat på *hur* användaren sveper fram sina ord och inte bara baserat på vilka ord användaren använder (som ordviktsförslaget ovan bygger på).

Maskininlärning används både av marknadsledarna inom just svepande gester för textinmatning (Swype och SlideIT) samt även inom andra forskningsområden som till exempel talanalys. Detta antyder att maskininlärning skulle kunna vara ett bra angreppssätt. Exakt hur stora förbättringar man skulle kunna uppnå är dock svårt att säga.

6.2 Jämförelse med SlideIT

Trots att algoritmerna i det här projektet i bästa fall slår SlideIT enligt den uppsatta testningsmetoden får man, oavsett vad dessa resultat säger, anta att SlideIT fungerar betydligt bättre för dagligt bruk. SlideIT har en mycket längre ordlista vilket ger bättre ordförslag i många situationer. Det här projektets algoritmer är begränsade av den relativt korta ordlistan som används för tillfället. SlideIT anpassar sig även efter användaren allt eftersom, vilket hela tiden kommer att förbättra ordförslagen.

Man får även komma ihåg att algoritmerna i det här projektet utvecklades utifrån testdata från endast fyra personer. Det finns säkert många olika sätt att svepa gester på som *inte* uppträder i denna testdata. Att välja endast tio testord från ett mycket stort urval är också en stor begränsning. SlideIT har däremot gått igenom mer produktutveckling och det har antagligen funnits mycket mer testdata att analysera. Således bör SlideIT kunna hantera dessa annorlunda gester betydligt mycket bättre.

6.3 Användbarhet

I det här projektet arbetades det enbart med att ta fram algoritmer som var så effektiva som möjligt på att översätta svepande gester till ord. Det som inte diskuterades är huruvida en så låg straffpoäng som möjligt faktiskt innebär högre användbarhet i verkliga tillämpningar. Projektets bästa algoritm föreslår, som bekant, långa ord först vilket fungerar bra i testningen. Det finns dock egentligen inte någon stark motivering till varför det skulle vara ett korrekt eller önskvärt beteende. Antagligen är det inte optimalt att bara sortera på långa ord även om testresultaten råkar visa det. Att till exempel använda en viktad ordlista skulle antagligen ge naturligare ordförslag till användaren.

Att alltid föreslå det vanligaste ordet är antagligen inte optimalt det heller. Ordförslag kan anses vara olika bra i olika sammanhang. Det är till exempel stor skillnad om användaren är 13 år och försöker skriva SMS-språk till sina jämnåriga kompisar, jämfört med en 46-årig tjänsteman som vill skriva korrekta mejl till sina kollegor på jobbet. Detta problem skulle kunna lösas med ordlistor riktade till olika målgrupper.

7 Slutsats

I det här projektet implementerades ett fungerande tangentbord som låter användaren skriva ord med hjälp av svepande gester. De bakomliggande algoritmerna som användes för att uttyda ord från dessa gester föreslog ord med en precision som i bästa fall var jämförbar med den hos marknadsledaren SlideIT.

Vi tycker att det är ganska förvånande att det gick att implementera algoritmer med så här enkla medel som ändå presterar bra. Vidare identifierades många möjliga sätt att ytterligare förbättra algoritmernas prestanda. Alltså drar vi slutsatsen att problemet som formulerades går att lösa mycket väl utan att ens behöva använda maskininlärning eller andra avancerade matematiska modeller.

8 Källförteckning

Dasur Pattern Recognition Ltd., (2012), *SlideIT Keyboard*, http://www.mobiletextinput.com, [2012-04-04]

Dasur Pattern Recognition Ltd., (2012), *SlideIT tangentbord*, https://play.google.com/store/apps/details?id=com.dasur.slideit, [2012-04-04]

Swype Inc., (2012), *Swype | Type Fast, Swype Faster*, http://www.swype.com, [2012-04-04]

FreePatentsOnline.com, (2012), *System and method for continuous stroke word-based text input,* http://www.freepatentsonline.com/7453439.html, [2012-04-04]

David Cardinal, (2011), *How does Swype really work?*, http://www.extremetech.com/extreme/97837-how-does-swype-really-work, [2012-04-04]

Rabiner, L. R., (1989), *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, 77 (2), pp 257 - 286

Wikipedia, (2012), *QWERTY*, http://en.wikipedia.org/wiki/Qwerty, [2012-04-04]

Wikipedia, (2012), *Levenshtein distance*, http://en.wikipedia.org/wiki/Levenshtein_distance, [2012-04-04]

Wikipedia, (2012), *Manhattan distance,* http://en.wikipedia.org/wiki/Manhattan_distance, [2012-04-04]

Steve Sinchak, (2011), *How to Touch the Windows 8 Metro Interface*, http://tweaks.com/windows/52231/how-to-touch-the-windows-8-metro-interface, [2012-03-15)

Krishna Bharadwaj, (2011), *How swype works?, http://krishnabharadwaj.info/how-swype-works*, [2012-03-14]

Appendix A: Testningsresultat

Algoritm	Straffpoäng	Rätt på första förslaget
Levenshtein Alg.	1460	35
Basic Alg. (non-strict)	1129	70
Basic Alg.	1037	71
Basic Alg. + Forgiving Filtering Alg.	1031	73
Basic Alg. + Long Words Alg.	495	257
Basic Alg. + Forgiving Filtering Alg. + Long Words Alg.	478	261
Beelining Alg.	379	247
Beelining Alg. + Forgiving Filtering Alg.	348	253
Beelining Alg. + Long Words Alg.	297	291
Beelining Alg. + Forgiving Filtering Alg. + Long Words Alg.	264	298
SlideIT	272	280

Appendix B: Urval av testningsutskrifter

R_α är de kandidater som föreslogs av den grundläggande algoritmen av den icke-strikta varianten.
R_β är de kandidater som föreslogs av den bästa kombinationen av algoritmer.

Ord	$ULS(G)$	R_α	R_β
ABSINTHE	ASDXCVBNBVFDSDFGTYUIKJNBHGTYHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDCVBVCXDSDFTYUIKJNBHGTHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCXDSASDFRTYUIKJNBHGTGHGFRE	ABASE, ABATE, ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCFDSDFRTYUIKJNBHYTYHJHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDXCVBVCXDSDFGTYUIKJNBHGTHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFGVBVCXDSDFGYUIKJNBHGTYHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASXCVBVCXDSDFGYUIKJNBHGTGHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCXDSDFGYUIKJNBHGTYHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFGVBVCXDSDFRTYUIKJNHGTHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCXDSDFGHYUIKJNBHGTGHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ATTACK	ASDRTREDSASDXCVHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTRDSASDXCVHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDFRTREDSASDXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASERTREDSASDXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTRDSASXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDFRTRDSASXCVHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDFRTRDSASXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDRTYTRDSASDXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTREDSASZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASWERTREDSZXCVHJKL	ALL, AWL	
BURNT	BHJUYTREDFGVBNMNJHT	BEET, BENT, BET, BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGVBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHYUYTRERGHJMNJHGTRT	BEET, BENT, BET, BUNT, **BURNT** ...	**BURNT** ...
BURNT	BJUYTRFGVBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRTGHBNHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGVBNMKJUYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRERFGVBNJHT	BEET, BENT, BET, BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGVBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
CORAL	CVGHJIOIUYTREDSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHORDAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORDAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKLOIUYTRESASDFGHJKL	CALL, CELL, CHILL, CHLORAL, CHORAL,	CHLORAL, CHORAL,

		CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJUIOIUYTREDSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHORDAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORDAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKLOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHLORAL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHLORAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CGHJIOPOIUYTREWASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, CORRAL, **CORAL** ...
CORAL	CGHJUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJKIOIUYTREDSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHORDAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORDAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJIOIUYTREDSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHORDAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORDAL, CHORAL, COITAL, CORRAL, **CORAL** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	KJHGFDSASDFGF		
LAG	LKJHGFDEWQASDFGH	LASH, LEASH	LEASH, LASH
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWQWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	PURR
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR,	**POWER** ...

		POUTER, **POWER** ...	
POWER	POIUYTREWQWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	PURR
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERT		QUEER
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTYU		**QUERY** ...
QUERY	QWERTYUYTREWERTY	**QUERY** ...	
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUIUYTREWERTY	**QUERY** ...	QUIET
WINNING	WERTYUIKMNJKIKMNJHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJUIKJNHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKMNJKIJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJUIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIOKMNJKIJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNJHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNJHGF		**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
ABSINTHE	ASDFCVBVCDSDFTYUIKJNBHGTYHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDXCVBVCDSDFGYUIOIKJNHGTGHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCDSDFTYUIKJNBHGTHBHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVFDSDFRTYUIJNBHGTGHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCFDSDFGYUIJNHGTGHGRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCXDSDFGHUIOIKJNBHGTGHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDXCVBVCDSDFGTYUIUJHBHYTYHGTRE	ABBE, ACE, ACHE, ACRE, ACUTE, AFIRE, AGE, AGREE, AGUE, ARE, ASSURE, ASTUTE, ATE, ATTIRE, AXE, AYE	
ABSINTHE	ASDFCVBVFDSDFGTYUIKJNHGTGHGRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFVBVCFDSDFGYUIKJNBHGTRTGHGFRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ABSINTHE	ASDFCVBVCXDSDFGTYUIJNBHGTGHGRE	ABBE, **ABSINTHE** ...	**ABSINTHE** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ATTACK	ASDRTREWSA.ZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTREDSA.ZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDFTYTREDSASZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDRTRDSASDXCVHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDRTRDSASZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...

ATTACK	ASDFRTREDSAZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTRESA.ZXCVBJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTREWSASZXCVBNJKL	ALL, AREAL, ASTRAL, AWL	ASTRAL
ATTACK	ASDERTREDSAZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTRDSASZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
BURNT	BHYUYTRFGVBNMNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNHT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGVBNJHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNBHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHJNHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNJHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUIUYTRGHNJHYT	BIGHT, BIT, BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNHYT	BUNT, **BURNT** ...	**BURNT** ...
CORAL	CVGHJUIOIUYTREWQASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJKLOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHLORAL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHLORAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREDSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHORDAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORDAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKIOIUYTREWSASDFGHJKL.L	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
POOL	POIKL	PILL, POL, POLL, **POOL** ...	PILL
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POLK		POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...

POOL	POL	POL, POLL, **POOL ...**	POLL, **POOL ...**
POOL	POL	POL, POLL, **POOL ...**	POLL, **POOL ...**
POOL	POL	POL, POLL, **POOL ...**	POLL, **POOL ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POUYTREWER	PEER, PER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER ...**	**POWER ...**
QUERY	QWERTYUIUYTREWERTY	**QUERY ...**	QUIET
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUIUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUIUYTRTY	QWERTY	QUIRT
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
QUERY	QWERTYUYTRERTY	**QUERY ...**	**QUERY ...**
WINNING	WERTYUIUJNJKIOIKJNBVGF	WOOF	**WINNING ...**
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIKJNJKIKJNBH		**WINNING ...**
WINNING	WERTYUIJNJKIKJNBVG	WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIKJNJIJNBHG	WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIKJNJUIUJNBHG	WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIJNBNJUIKJNBHG	WEBBING, WIG, WING, **WINNING ...**	**WINNING ...**
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING ...**	**WINNING ...**
ABSINTHE	ASDCVBVCFDSDFGHYUIKMNJHYTGHGTRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDFCVBVFDSDFGHJUIKJNHGTGHGFRE	ABBE, ABJURE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDFCVBVFDSDFGHJUIKMNHYTHGTRE	ABBE, ABJURE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDFCVBVCFDSDFGHJKIKMNJHYTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDXCVBVCDSDFGHJKIKMNJHGTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASXCVBVFDSDFGHJUIKJNBHGTGHGTRE	ABBE, ABJURE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDFCVBVCFDSDFGHJUIKJNJHYTGHGTRE	ABBE, ABJURE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDCVBVCDSDFGHJUIKJNHGTGHGRE	ABBE, ABJURE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASXCVBVFDSDFGHJUIKJNJHYTGHGRE	ABBE, ABJURE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDCVBVCFDSDFGTYUIKMNHYTGHGTRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**

ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ADD	ASD	AD, **ADD** ...	**ADD** ...
ATTACK	ASDERTREWQA.ZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTREDSASZXCVBNMK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDFGTREWQA.ZXCVGHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTREWQASZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASERTREWSA.ZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDRTRDSASZXCVGHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTREWSAZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDRTREDSAZXCVGHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDFRTRDSAZXCVGHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
ATTACK	ASDERTRFDSA.ZXCVBHJK	ARK, ASK, **ATTACK** ...	**ATTACK** ...
BURNT	BHJUYTRFGVBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHYUYTRFGVBNHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHYUYTRFGVBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHYUYTRFGHBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGVBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGVBNJHGT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHJUYTRFGHBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
BURNT	BHYUYTRFGHBNJHYT	BUNT, **BURNT** ...	**BURNT** ...
CORAL	CGHJUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWQASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJKIOIUYTREWQASDFGHJKL.		CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKIOIUYTRESASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJGFDSASDFG	**LAG** ...	**LAG** ...
LAG	KJHGFDSASDFG		
LAG	LKJHGFDSASDFGH	LASH	LASH

LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFGH	LASH	LASH
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWERT	PERT, PET, PIT, POET, PORT, POT, POUT, PUT, PUTT	POET, PUTT, PET
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWQWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	PURR
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTYU		**QUERY** ...
QUERY	QWERTYUYTRTY	QWERTY	
WINNING	WERTYUIKJNMKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNKIKMNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIUJNJIUJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNJHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIUJNJUIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIOIKJNBG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBVG	WIG, WING, **WINNING** ...	**WINNING** ...

WINNING	WERTYUIKJNJKIKMNHG	WIG, WING, **WINNING ...**	**WINNING ...**
ABSINTHE	ASDXCVBVCFDSDFGTYUIKMNBHGTGHGFDEW	ANEW, ASKEW, AW	**ABSINTHE ...**
ABSINTHE	ASDXCVBVCFDSDFGHYUIKJNHYTYHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDXCVBNBVGFDSDFRTYUIKMNBHGTYHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDXCVBVGFDSDFRTYUIKJNBHGTRTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDXCVBNBVCFDFGHYUIKMNBHGTGHGFRE	ABBE, ACE, ACHE, ACME, ACNE, ACRE, ACUTE, AFIRE, AGE, AGREE, AGUE, ANTE, ARE, ASSUME, ASSURE, ATE, AXE, AYE	ASS
ABSINTHE	ASDXCVBVCFDSDFRTYUIKJNBHGTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDXCVBVGFDSDFGTYUIKJNBHGTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASZXCVBNBVCFDSDFGTYUIKJNBHGTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDXCVBVCFDSDFRTYUIKJNBHGFRTGHGFRE	ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ABSINTHE	ASDFCVBHGFDSASDFRTYUIKJNBHGFRTGHGFRE	ABASE, ABATE, ABBE, **ABSINTHE ...**	**ABSINTHE ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASDFD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ADD	ASD	AD, **ADD ...**	**ADD ...**
ATTACK	ASDERTREDSASZXCVBNJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFRTREWSASZXCVBHJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFTYTREDSA.ZXCVBNMK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFGTREDSA.ZXCVBNJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFRTREDSA.ZXCVBNJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFRTREWSASZXCVBHJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDERTREWSAZXCVBNJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFRTREDSA.ZXCVBNJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDFGTREWSA.ZXCVBJK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
ATTACK	ASDERTREWSA.ZXCVBNK	ARK, ASK, **ATTACK ...**	**ATTACK ...**
BURNT	BNJUYTRFGHBNJHGT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTRFGHBNBHGT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTREDFGVBNJHGT	BEET, BENT, BET, BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTREDFGVBNJHYT	BEET, BENT, BET, BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTRFGHBNJHYT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BNJUYTRFGHBNBHGT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTRFGVBNJHGT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTRFGHBNHGT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTRFGHBNBHGT	BUNT, **BURNT ...**	**BURNT ...**
BURNT	BHJUYTRERFGVBNBHGT	BEET, BENT, BET, BUNT, **BURNT ...**	**BURNT ...**
CORAL	CVBHJKIOIUYTREDSDFGHJKL	CELL, CHILL, CHURL, COIL, COOL, CREEL, CULL, CURL	COO
CORAL	CVBHJKLOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHLORAL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL ...**	CHLORAL, CHORAL, COITAL, CORRAL, **CORAL ...**
CORAL	CVGHJUIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL ...**	CHORAL, COITAL, CORRAL, **CORAL ...**
CORAL	CVBHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL ...**	CHORAL, COITAL, CORRAL, **CORAL ...**
CORAL	CVBHJKIOIUYTREDSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHORDAL,	CHORDAL, CHORAL,

		CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWSDFGHJKL	CELL, CHILL, CHURL, COIL, COOL, COWL, CREEL, CULL, CURL	COWL
CORAL	CVBHJKLOIUYTREWSASDFGHJKL	CALL, CELL, CHILL, CHLORAL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHLORAL, CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVGHJKIOIUYTREWQASDFGHJKL	CALL, CELL, CHILL, CHORAL, CHURL, COAL, COIL, COITAL, COOL, **CORAL** ...	CHORAL, COITAL, CORRAL, **CORAL** ...
CORAL	CVBHJKIOIUYTREWSDFGHJKL	CELL, CHILL, CHURL, COIL, COOL, COWL, CREEL, CULL, CURL	COWL
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
LAG	LKJHGFDSASDFG	**LAG** ...	**LAG** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POOL	POL	POL, POLL, **POOL** ...	POLL, **POOL** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWERT	PERT, PET, PIT, POET, PORT, POT, POUT, PUT, PUTT	POET, PUTT, PET
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWERTR	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	POET, PUTT, PET
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
POWER	POIUYTREWER	PEER, PER, PIER, POOR, POTTER, POUR, POUTER, **POWER** ...	**POWER** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...

QUERY	QWERTYUYTREWERTY	**QUERY** ...	
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
QUERY	QWERTYUYTRERTY	**QUERY** ...	**QUERY** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNBNJKIKMNBHG	WEBBING, WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKMNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJUIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKMNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKJNBHG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNJKIKMNBVG	WIG, WING, **WINNING** ...	**WINNING** ...
WINNING	WERTYUIKJNBHJUIUJNBVG	WEBBING, WIG, WING, **WINNING** ...	**WINNING** ...