Tastaturbelegung optimieren

Andreas Wettstein
wettstae@gmail.com
April 2019

1 Vorbereitung: Programm übersetzen

Um den Optimierer benutzen zu können, muss er übersetzt werden. Dabei wird der für Menschen lesbare Programmtext in eine Form überführt, in der sie der Computer direkt ausführen kann. Zum Übersetzen braucht man einen C++-Compiler, zum Beispiel den kostenlosen GNU C++-Compiler (ab Version 4.8.1). Damit geht das Übersetzen so:

```
g++ -std=c++11 -O2 -DNDEBUG opt.cc -o opt
```

Resultat ist das ausführbare Programm `opt`. Für den ebenfalls kostenlosen LLVM C++-Compiler (ab Version 3.3) lautet das Kommando zum Übersetzen

```
clang++ -stdlib=libstdc++ -std=c++11 -O2 -DNDEBUG opt.cc -o opt
```

Der Optimierer kann für eine bestimmte Zahl von Tasten übersetzt werden, zum Beispiel mit

```
g++ -std=c++11 -O2 -DNDEBUG -DTASTENZAHL=32 opt.cc -o opt
```

für 32 Tasten. Die Voreinstellung ist 35 Tasten, dabei sind zwei Shifttasten mitgezählt. Wenn man die Tastenzahl ändert muss man auch die Konfiguration ändern, siehe Abschnitt 6.

Um die bestmögliche Geschwindigkeit zu erzielen, kann man mit den Optionen des Compilers experimentieren, zum Beispiel

```
g++ -std=c++11 -Wall -Ofast -DNDEBUG -DOHNE2SHIFT \
    -DMIT_THREADS -pthread opt.cc -o opt
```

Hier bewirkt die Option `-DOHNE2SHIFT`, dass Grossbuchstaben, die als zweites in einer Folge zweier Zeichen auftreten, nicht berücksichtigt werden. Da Grossbuchstaben fast nur am Wortanfang (also als erstes Zeichen) vorkommen, hat diese Vereinfachung meist keinen Einfluss auf das Resultat.

Wenn der Optimierer Multithreading (die Verwendung mehrerer Prozessorkerne gleichzeitig) unterstützen soll, muss man beim Übersetzen die Option `-DMIT_THREADS` angeben. Auf vielen Systemen sind weitere Optionen nötig. Zum Beispiel brauche ich auf meinem System die Option `-pthread`.

Normalerweise benutzt der Optimierer die Zeichencodierung UTF-8. Mit der `-DAUSGABE_8BIT` Compileroption kann man die Terminalausgabe auf ISO-8859-1 umstellen.

2 Was man tippen will: Korpus und Häufigkeitsfiles

Um mit Computerhilfe eine gute Tastaturbelegung zu finden, muss man angeben, was man tippen will. Dazu benutzt man ein *Korpus*, eine Textsammlung, die die zu tippenden Texte repräsentiert.

Der Optimierer benutzt nur einen Teil der Information im Korpus: die Zeichenhäufigkeiten, die Häufigkeiten von Zeichenpaaren (*Bigrammen*) und Zeichentripeln (*Trigrammen*). Diese Häufigkeiten sind in Files mit gleichem Namensstamm und den Endungen `.1`, `.2` und `.3` abgelegt. Bigramme und Trigramme überlappen. Zum Beispiel enthält das Wort «Velo» die Bigramme «Ve», «el» und «lo» sowie die Trigramme «Vel» und «elo».

Beim Optimierer sind Häufigkeitsfiles für Deutsch und für Englisch dabei. Bei den Files mit `-t` im Namen wurden nur Bi- und Trigramme mitgezählt, die im Innern von Worten liegen und keine Trennstelle enthalten.

2.1 Eigene Häufigkeitsfiles erstellen

Um eine eigene Textsammlung zu benutzen, geht man so vor:

1. Man packt die Sammlung in ein File, zum Beispiel `meinkorpus.txt`.
2. Man stellt sicher, dass das File UTF-8-codiert ist.
3. Man benutzt `opt`, um die Häufigkeitsfiles zu erzeugen. Zum Beispiel erzeugt

   ```
   ./opt meinkorpus.txt
   ```

   aus der Textsammlung im File `meinkorpus.txt` die drei Häufigkeitsfiles `meinkorpus.txt.1`, `meinkorpus.txt.2` und `meinkorpus.txt.3` sowie eine Wortliste in `meinkorpus.txt.wl`.

Wenn man `opt` mit zwei Argumenten aufruft wird angenommen, dass das erste Argument ein File mit UTF-8-codierten TEX-Trennmustern (`*.pat.txt`) ist, das zweite die Textsammlung. Es werden Häufigkeitsfiles erzeugt, bei denen nur Bigramme und Trigramme mitgezählt sind, die im Innern von Worten liegen und keine Trennstelle enthalten, und statt der Wortliste wird eine Silbenliste gebildet. Zum Beispiel:

```
./opt hyph-de-1996.pat.txt meinkorpus.txt
```

`opt` kann auch Häufigkeitsfiles zusammenzählen. Dazu ruft man es mit mehr als zwei Argumenten auf. Zum Beispiel erzeugt

```
./opt deutsch.txt deutsch-t.txt deutsch-t.txt gemischt.txt
```

aus den mitgelieferten Häufigkeitsfiles für Deutsch neue Files, bei denen Bi- und Trigramme an Wortgrenzen und mit Trennstellen nur zu einem Drittel gezählt werden. Eine andere Möglichkeit, Häufigkeitsfiles zu kombinieren und zu gewichten, ist die Option `-G`, siehe unten.

2.2 Häufigkeitsfiles benutzen

Beim Start des Optimierers gibt man den Namensstamm der Häufigkeitsfiles an. Dazu benutzt man die Optionen `-2` oder `-3`. Mit `-2` werden nur die Zeichen- und Bigrammhäufigkeiten benutzt, mit `-3` auch die Trigrammhäufigkeiten. Zum Beispiel startet

```
./opt -2 deutsch.txt
```

eine Optimierung mit den Häufigkeitsfiles für Deutsch, bei der Trigramme ignoriert werden.

Falls das Häufigkeitsfile mit den Zeichenhäufigkeiten (im Beispiel oben `deutsch.txt.1`) nicht existiert, versucht der Optimierer, ein File mit dem angegebenen Namen (also `deutsch.txt`) zu öffnen. Gelingt das, wird das File als Korpus behandelt. Man kann so den Zwischenschritt über Häufigkeitsfiles überspringen, jedoch dauert das Einlesen des Korpus länger.

Man kann `-2` oder `-3` mehrfach benutzen. Die so angegeben Häufigkeitsfiles werden gleich gewichtet, unabhängig von der Grösse der tabellierten Korpora. Mit der Option `-G` (Voreinstellung 1) kann man die Gewichtung nachfolgend angegebener Häufigkeitsfiles ändern, siehe Gleichung (1). Zum Beispiel führt

```
./opt -2 deutsch.txt -G 3 -2 englisch.txt
```

eine Optimierung durch, in der Englisch dreimal so stark gewichtet wird wie Deutsch.

Die separate Angabe verschiedener Häufigkeitsfiles durch mehrfaches `-2` oder `-3` ist etwas rechenaufwändiger, als ein einziges Summenkorpus zu verwenden. Dem steht der Vorteil gegenüber, dass das Gesamt-Optimum (wenn man es denn findet) Pareto-optimal bezüglich der einzelnen Korpora ist: Man kann die Belegung für kein Korpus verbessern, ohne sie für einen andern zu verschlechtern. Wenn man ein Summenkorpus verwendet, ist das nicht garantiert (siehe Abschnitt 7.3).

2.3 Korpusgrösse, systematischer und statistischer Fehler

Wenn man statt Häufigkeitsfiles direkt ein volles Korpus einliest, wird bei der Bewertung von Belegungen aus einem File (siehe Option `-r` in Abschnitt 5) die Standardabweichung des Aufwands und die Standardabweichung der Differenz zum Aufwand der ersten Belegung im File geschätzt und ausgegeben. Dafür wird angenommen, dass die Häufigkeiten von verschiedenen Wörtern in einem Text unkorreliert sind. Ein «Wort» ist hier eine Zeichenfolge, die durch ein Leerzeichen oder ein Zeichen, das in der Belegung nicht vorkommt (siehe Abschnitt 6.1) begrenzt wird. Für die Häufigkeiten von Wörtern wird eine Binomialverteilung angenommen, deren Mittelwert und Varianz aus den relativen Häufigkeiten im Korpus geschätzt werden. Bei der Berechnung der Standardabweichung werden Trigramme und fehlende Zeichen nicht berücksichtigt.

Aufwandsdifferenzen von einer Standardabweichung oder weniger sind unerheblich, das heisst, sie könnten ohne weiteres allein durch die zufällige Textauswahl zustande kommen. Unterschiede von drei Standardabweichungen oder mehr hingegen darf man als real ansehen. Um die Standardabweichung zu reduzieren, muss man das Korpus vergrössern. Die Standardabweichung sinkt mit der Quadratwurzel der Korpusgrösse. Um zum Beispiel die Standardabweichung zu halbieren, muss man ein viermal grösseres Korpus benutzen.

Der statistische Fehler ist nicht die einzige Unsicherheit, die vom Korpus herkommt. Zum Beispiel hat die Art der Texte (Texte mit langen oder kurzen Sätzen, mit vielen oder wenigen Fremdwörtern) einen systematischen Einfluss, der sich in der Standardabweichung nicht zeigt. Nach meiner Erfahrung ist ein Korpus von wenigen Megabyte ausreichend gross.

2.4 Zeichen durch andere ersetzen

Will man ein einzelnes Zeichen durch Tippen einer Folge von einem oder mehreren anderen Zeichen eingeben, kann man das in der Konfiguration mit `Ersatz` erreichen, siehe Abschnitt 6.1. Es ist nicht notwendig, das Korpus oder die Häufigkeitsfiles zu verändern.

Will man hingegen eine Folge von mehreren Zeichen durch eine andere Folge von Zeichen eingeben, funktioniert dieses einfache Verfahren nicht, da in den Häufigkeitsfiles nicht genug Information steckt, um allgemeine Ersetzungen zu machen. In diesem Fall muss man im Korpus die zu erzeugende Zeichenfolge durch die zu tippende ersetzen. Aus diesem veränderten Korpus erzeugt man neue Häufigkeitsfiles.

2.5 Weitere Operationen

Mit der Option `-T` kann man aus einem Korpus und einem File mit Trennmustern ein neues Korpus erzeugen, in dem alle möglichen Trennstellen durch weiche Trennzeichen (U+00AD) markiert sind. Zum Beispiel:

```
./opt -T hyph-de-1996.pat.txt eingabe.txt ergebnis.txt
```

Bigramm	Tasten	Tastenbi- und trigramme
xy	$t_x t_y$	$t_x t_y$
Xy	$s_x t_x t_y$	$s_x t_x t_y$, $t_x t_y$, $s_x t_x$
xY	$t_x s_y t_y$	$t_x s_y t_y$, $s_y t_y$, $t_x s_y$
XY	$s_x t_x s_y t_y$	$s_x t_x s_y$, $t_x s_y t_y$, $s_x t_x$, $t_x s_y$, $s_y t_y$

Tabelle 1: Zerlegung von Bigrammen in Tastenbi- und trigramme. t_X ist die Taste für x, s_X die zugehörige Shifttaste, t_Y und s_Y analog. $s_X t_X$ und $s_Y t_Y$ zählen als Einzeltasten, da sie Grossbuchstaben entsprechen. $t_X t_Y$ und $t_X s_Y$ und sind Tastenbigramme, $s_X t_X t_Y$ und $s_X t_X s_Y$ sind Shift-Bigramme, $t_X s_Y t_Y$ ein Tastentrigamm.

3 Das Bewertungsschema

Um eine Belegung zu bewerten, betrachtet man die Tastenanschläge, die nötig sind, um das Korpus damit einzugeben. Wie beim Korpus beschränkt man sich bei der Bewertung auf einige Kriterien. Jedes dieser Kriterien steuert einen Beitrag zum *Aufwand* bei. Die Summe dieser Beiträge ist der Gesamtaufwand, siehe Gleichung (2).

3.1 Mechanische Kriterien

Einzeltastenaufwände. Jede Taste hat einen Aufwand. Dieser wird mit der Häufigkeit multipliziert, mit der sie angeschlagen wird, also mit der Häufigkeit des Zeichens, mit dem die Taste belegt ist. Die Summe dieser Produkte ist der *Lageaufwand*, siehe Gleichung (3).

Bigrammaufwände. Jede Folge zweier Tasten hat einen Aufwand, den Bigrammaufwand. Wir nennen hier die Folge zweier Tastenanschläge *Bigramm*, genauso wie ein Zeichenpaar. Der Klarheit wegen werden wir zum Teil *Tastenbigramm* schreiben. Der Unterschied wird in Tabelle 1 durch Zerlegung des Bigramms «xy» mit allen Kombinationen von Gross- und Kleinschreibung gezeigt. Dabei treten Tastenbigramme der Form Shift+Zeichentaste auf, deren Häufigkeit aus Zeichenhäufigkeit bestimmt wird und deren Aufwände zum Lageaufwand zählen. Ferner treten normale Tastenbigramme aus zwei Zeichentasten, Shift-Bigramme und Trigramme auf, so dass der Beitrag durch Bigramme recht komplex wird, siehe Gleichung (6).

Shift-Bigramme bestehen aus einer Shifttaste, der zugehörigen Zeichentaste und der darauf folgenden Taste. Zu ihrer Bewertung wird der Bigrammaufwand für die Shifttaste und die letzte Taste mit einem Faktor multipliziert, siehe Gleichung (6).

Trigrammaufwände. Einer Folge von drei Tastendrücken wird ein Aufwand zugewiesen, der Trigrammaufwand. Nur wenn man Option `-3` verwendet werden alle Tastentrigramme bewertet, ansonsten nur solche, deren Häufigkeit bereits durch Bigrammhäufigkeiten festgelegt ist, siehe Tabelle 2 und Gleichung (7).

Fingerbelastung. Für jeden Finger wird eine *Zielhäufigkeit* festgelegt, also der Anteil an den Anschlägen, die er tippen soll. Wenn die tatsächliche Häufigkeit die Zielhäufigkeit überschreitet,

Trigramm	Tasten	Tastentrigramme
xyz	$t_x t_y t_z$	$t_x t_y t_z$
Xyz	$s_x t_x t_y t_z$	$s_x t_x t_y$, $t_x t_y t_z$
xYz	$t_x s_y t_y t_z$	$t_x s_y t_y$, $s_y t_y t_z$
XYz	$s_x t_x s_y t_y t_z$	$s_x t_x s_y$, $t_x s_y t_y$, $s_y t_y t_z$
xyZ	$t_x t_y s_z t_z$	$t_x t_y s_z$, $t_y s_z t_z$
XyZ	$s_x t_x t_y s_z t_z$	$s_x t_x t_y$, $t_x t_y s_z$, $t_y s_z t_z$
xYZ	$t_x s_y t_y s_z t_z$	$t_x s_y t_y$, $s_y t_y s_z$, $t_y s_z t_z$
XYZ	$s_x t_x s_y t_y s_z t_z$	$s_x t_x s_y$, $t_x s_y t_y$, $s_y t_y s_z$, $t_y s_z t_z$

Tabelle 2: Zerlegung von Trigrammen in Tastentrigramme. t_X ist die Taste für x, s_X die zugehörige Shifttaste, t_Y, s_Y, t_Z und s_Z analog. Nur für die Häufigkeiten von $t_X t_Y t_Z$ und $t_X t_Y s_Z$ brauchen wir Trigrammenhäufigkeiten. Für die anderen Tastentrigramme und Shift-Bigramme genügen Bigrammhäufigkeiten.

wird der Überschuss quadriert und mit einem Gewicht multipliziert, um einen Aufwand zu erhalten, siehe Gleichung (8). Ausgenommen sind *fixe Finger*: Finger, deren Anschläge nicht von der Belegung abhängen, sondern durch Konfiguration und Korpus festgelegt sind.

3.2 Nichtmechanische Kriterien

Die bisher beschriebene Bewertung beschäftigt sich mit der Mechanik des Tippens. Möglicherweise hängt die Häufigkeit, mit der man ähnliche Zeichen verwechselt, davon ab, wie die entsprechenden Tasten zueinander liegen. Im Konfigurationsfile kann man den Grad der Ähnlichkeit von Zeichen als Zahl angeben. Diese wird mit dem *Verwechslungspotenzial* multipliziert, um einen Aufwand zu bekommen, siehe Gleichung (9). Das Verwechslungspotenzial hängt von den Tasten ab, denen die Zeichen zugeordnet sind und lässt sich im Konfigurationsfile einstellen. In der Voreinstellung sind alle Zeichen unähnlich.

Manche Anwender wollen bestimmte Zeichen in einem bestimmten Teil der Belegung haben. Diese *Vorlieben* für die Zuordnung von Zeichen zu Tasten kann man im Konfigurationsfile mit einer Zahl festlegen, die das Ausmass der Vorliebe angibt. Jede erfüllte Vorliebe reduziert den Aufwand um diese Zahl, siehe Gleichung (9).

Auch durch Manipulation von Korpus und Häufigkeitsfiles kann man die Bewertung zu beeinflussen. Zum Beispiel wurde in einem Beitrag auf der Neo-Mailingliste behauptet, dass guter Schreibfluss innerhalb von Silben wichtiger ist als über Silbengrenzen hinweg. Verwendet man zur Optimierung Häufigkeitsfiles, bei denen nur Bi- und Trigramme mitgezählt sind die keinen Trennstelle enthalten (siehe Abschnitt 2), kann man dem näherungsweise Rechnung tragen (näherungsweise, weil Silbengrenzen und Trennstellen nicht ganz dasselbe sind).

Zum Vergleich von Belegungen mit unterschiedlicher Zeichenausstattung kann man einen Aufwand für die Zeichen veranschlagen, die sich mit einer Belegung nicht eingeben lassen.

3.3 Bewertungskriterien ausgeben

Mit der Option `-A` werden alle von Null verschiedenen Bigrammaufwände, Verwechslungspotenziale und Vorlieben ausgegeben. Mit `-3` werden auch die Trigrammaufwände ausgegeben.

4 Der Ablauf einer Optimierung

Der Optimierer benutzt ein einfaches Verfahren, das von einer zufälligen Belegung ausgeht. Diese wird durch wiederholtes Vertauschen von Zeichen schrittweise verbessert. Falls keine Vertauschung mehr eine Verbesserung bringt, ist man am Ende. Das Resultat ist eine *lokal optimale* Belegung.

Eine lokal optimale Belegung kann weit davon entfernt sein, optimal im Sinne der Bewertungskriterien zu sein. Daher wird das obige Verfahren vielfach (mit immer neuen Ausgangsbelegungen) durchgeführt. Die Zahl der Wiederholungen kann mit der Option `-i` angegeben werden. Ohne diese Angabe läuft der Optimierer bis er abgebrochen wird. Wie viele Wiederholungen man durchführen muss, hängt von den Bewertungskriterien, vielleicht auch vom Korpus ab. Nach meiner Erfahrung ist 10000 für die vorgegebenen Kriterien ein vernünftiger Wert, wenn man die Option `-2` verwendet. Mit Option `-3` braucht man eher mehr Wiederholungen.

Normalerweise wird jede lokal optimale Belegung ausgegeben, die besser als alle bisher gefundenen ist. Wenn man mit der Option `-m` eine Schwelle angibt, wird jede lokal optimale Belegung angezeigt, deren Aufwand unterhalb dieser Schwelle liegt.

Mit der Option `-s` kann man eine positive, ganze Zahl für den Saatwert des Zufallsgenerators angeben. Das kann nützlich sein, um reproduzierbare Optimierungsläufe zu erhalten. Ohne `-s` wird der Saatwert zufällig gewählt.

Die Zahl der Threads, die verwendet werden sollen, kann man mit `-t` angeben, sonst wird ein Thread verwendet. Die mit `-i` angegebene Zahl der Wiederholungen versteht sich pro Thread. Wenn man mehr Threads verwendet kann man diese Zahl also entsprechend senken.

5 Die Ausgabe des Optimierers

Zunächst einige Begriffe: *Handwechsel* sind Bigramme, deren Tasten von verschiedenen Händen angeschlagen werden. *Doppeltanschläge* sind Bigramme, bei eine Taste zweimal angeschlagen wird. *Kollisionen* sind Bigramme, bei denen verschiedene Tasten vom selben Finger angeschlagen werden. *Nachbaranschläge* sind Bigramme, deren Tasten von benachbarten Fingern derselben Hand angeschlagen werden. *Einwärtsbewegungen* sind Bigramme, bei denen die erste Taste von einem Finger weiter aussen an derselben Hand als die zweite angeschlagen wird (Der äusserste Finger ist der kleine Finger, der innerste der Zeigefinger). *Auswärtsbewegungen* verlaufen umgekehrt. Der Vorsatz «Shift-» kennzeichnet besondere Bigramme, deren erste Taste Shift und deren zweite Taste eine Zeichentaste ist.

5.1 Textausgabe

Hier die Ausgabe für «Aus der Neo-Welt», bewertet mit einem deutsch-englisch gemischten Korpus:

```
 Aus der Neo-Welt 382.859 Gesamtaufwand  187.075 Lageaufwand        links rechts
                    1.029 Kollisionen      6.976 Shift-Kollisionen  ob  5.7 11.8
   kuü.ä vgcljf    71.404 Handwechsel     24.118 Shift-Handwechsel  mi 36.4 32.1
   hieao dtrnsß     1.796 Ein-/Auswärts   25.117 Ein- oder auswärts un  5.2  8.9
   xyö,q bpwmz      9.262 benachbart      22.116 Shift-benachbart  sum 47.2 52.8
                   8.4 11.2 14.0 13.7 --.- --.- 17.6 10.8 14.3 10.1 Sh  2.9  1.2
```

Links oben steht der Name der Belegung oder eine Laufnummer. Daneben stehen Gesamt- und Lageaufwand. Links ist die Belegung gezeigt, daneben Bigrammhäufigkeiten: der Prozentsatz aller normalen Tastenbigramme (ohne solche, die ein Leerzeichen enthalten), die Kollisionen, Handwechsel oder Nachbaranschläge sind, sowie das Verhältnis von Ein- zu Auswärtsbewegungen. In der Spalte rechts daneben steht der Prozentsatz von Shift-Bigrammen, die Kollisionen, Handwechsel oder Nachbaranschläge sind. Rechts stehen die Anteile, mit denen sich die Anschläge auf die drei Tastenzeilen und die Hände verteilen. Die unterste Zeile zeigt die Verteilung der Anschläge auf die Finger (von linkem zu rechtem Kleinfinger) und die Anschlagshäufigkeiten der Shifttasten. Das `--.-` für die Daumen kommt daher, dass die Leertaste einem Daumen zugeordnet wurde, aber keinem bestimmten. Daher lassen sich für die einzelnen Daumen keine Häufigkeiten angeben. Anschläge die auf eine unbestimmte Daumentaste fallen werden zwar im Gesamt- und Lageaufwand berücksichtigt, aber für die restliche Ausgabe nicht mitgezählt.

Verwendet man Option `-3`, bekommt man zwei zusätzliche Zeilen:

```
4.765 kein Handwechs. 44.851 zwei Handwechsel
3.582 Wippe            6.403 IndirKollision
```

Die erste gibt die Häufigkeiten von Trigrammen ohne und mit zwei Handwechseln an, die zweite die Häufigkeit von Trigrammen, die aus einer Ein- und einer Auswärtsbewegung bestehen (Reihenfolge egal), und die von Trigrammen mit zwei Handwechseln bei denen die erste und letzte Taste verschieden sind, aber vom selben Finger getippt werden (*indirekte Kollisionen*).

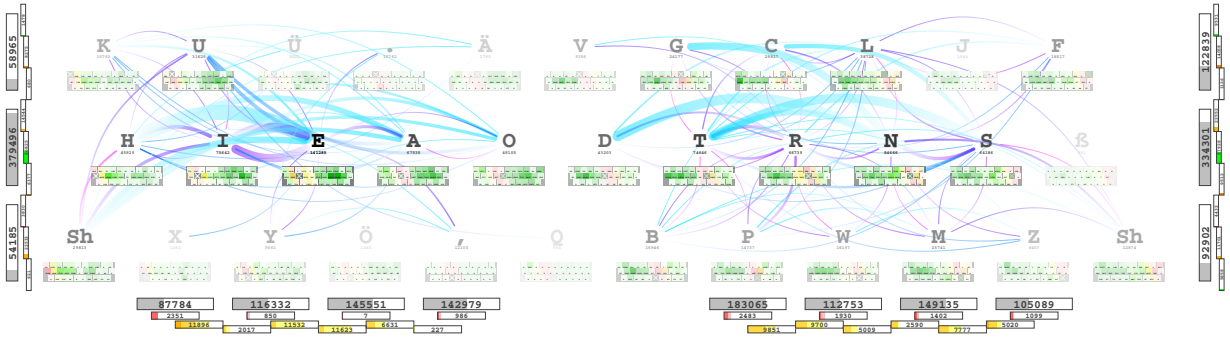


Abbildung 1: Beispiel für eine Belegungsgrafik

Mit der Option `-b`, gefolgt von einer Zahl zwischen 0 und 100, erhält man eine ausführliche Beschreibung der häufigsten Bigramme ohne Handwechsel. Zudem wird die Häufigkeit von Kollisionen und Nachbaranschlägen pro Finger beziehungsweise Fingerpaar aufgeschlüsselt. Verwendet man `-b` ein zweites Mal (gefolgt von einer Zahl) bekommt man zusätzlich Shift-Bigramme, verwendet man sie ein drittes Mal auch noch Trigramme.

Option `-k` verkürzt die Ausgabe auf eine Zeile pro Belegung. Mit der Option `-m` kann man einen Gesamtaufwand angeben, unterhalb dem alle lokale optimalen Belegung ausgegeben werden. Sonst wird immer nur die aktuell beste Belegung angezeigt.

Mit der Option `-w` kann man ein File mit einer Wortliste angeben. Das File muss UTF-8-codiert sein. In jeder Zeile muss eine Worthäufigkeit und, durch Leerzeichen getrennt, das entsprechende Wort stehen. Für eine gegebene Belegung werden alle Wörter in Abschnitte unterteilt, die mit derselben Hand getippt werden. Die Häufigkeiten dieser *Handeinsätze* werden gesammelt, aufsummiert und nach Häufigkeit sortiert ausgegeben. Normalerweise werden alle Handeinsätze aufgeführt, deren Häufigkeit insgesamt 95 % aller Handeinsätze ausmacht. Diese Grenze kann man mit der Option `-H` ändern.

Mit der Option `-r` kann man ein File angeben, in dem Belegungen stehen, die bewertet ausgegeben werden sollen, ohne dass eine Optimierung durchgeführt wird. Zusätzlich kann man mit Option `-V` diese Belegungen variieren. Von jeder Belegung werden alle Variationen erzeugt, die für bis zu der mit `-V` angegeben Zahl von Tasten von der gegebenen abweichen. Ist im File ein Zeichen einer Belegung als Grossbuchstabe angegeben, wird seine Position nicht variiert. Man sollte die Ausgabe mit `-m` auf die besseren Variationen einschränken, da sonst die Zahl der Belegungen schnell unhandhabbar gross wird.

5.2 Grafische Ausgabe

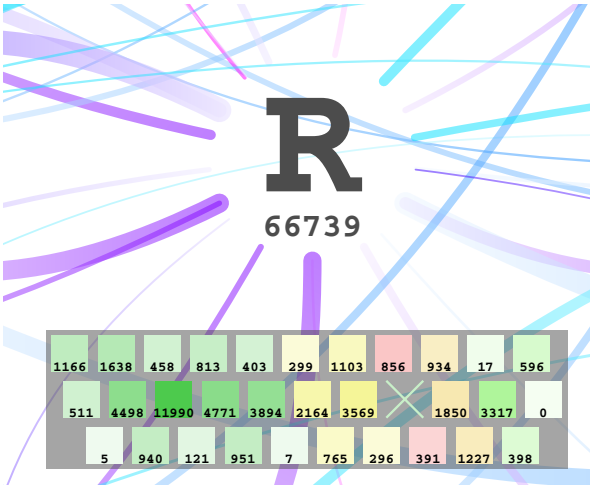
Mit der Option `-g` kann man den Namen eines PostScript-Files angeben, in das Grafiken für alle Belegungen geschrieben werden, die auch als Text ausgegeben werden. Zum Beispiel erzeugt

```
./opt -2 deutsch.txt -r bsptast.txt -g bsptast.ps
```

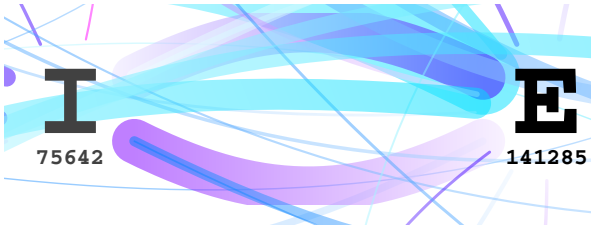
Grafiken in `bsptast.ps`, die die Belegungen in `bsptast.txt` für deutsche Texte auswerten. Abbildung 1 zeigt ein Beispiel.[1]

Die Grafiken haben zwei Ebenen. Der Vordergrund zeigt die Zeichen auf den Tasten, die Schwärze entspricht ihrer Häufigkeit. Diese steht als Zahlenwert unter den Zeichen (10000 entspricht 1 %).

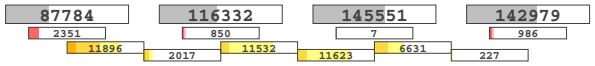
[1] Für die Abbildungen wurde das PostScript-File in PDF umgewandelt. Dadurch steigt die Filegrösse, dafür beherrscht PDF im Gegensatz zu PostScript Transparenz.



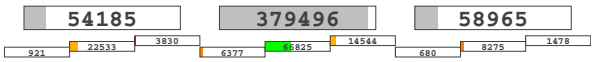
(a) Vordergrund: Einzelzeichen und folgende Anschläge



(b) Hintergrund: Bögen für Bigramme



(c) Unterer Rand: Fingerbelastung



(d) Seitenrand (gedreht): Gruppen und Zeilensprünge

Abbildung 2: Details der Belegungsgrafik

Darunter zeigt eine Minitastatur die Verteilung der folgenden Anschläge. Die Farbe ist je nach Bigrammtyp verschieden, die Intensität und Zahlen auf den Minitasten geben die Häufigkeit der Bigramme an. Abbildung 2a zeigt ein Beispiel. Man erkennt, dass auf «r» und «R» etwa 6,7 % der Anschläge entfallen und dass nach einem «r» oder «R» am häufigsten die Taste in der Grundposition des linken Mittelfingers angeschlagen wird. Sie ist im Beispiel mit «e» belegt, und etwa 1,2 % aller Bigramme sind r-e.

Der Hintergrund zeigt Tastenbigramme durch Bögen. Die Bogendicke entspricht der Häufigkeit, die Farbe ist je nach Bigrammtyp verschieden. Die Reihenfolge, in der die Tasten eines Bigramms angeschlagen werden, wird durch den Intensitätsverlauf und die Krümmung angezeigt. Für die rechte Hand läuft die Bewegung gegen den Uhrzeigersinn, für die linke im Uhrzeigersinn. Seltene Bigramme und Handwechsel sind der Übersichtlichkeit wegen weggelassen. Abbildung 2b zeigt dicke violette Bögen, die «I» und «E» verbinden. Im Beispiel werden «I» und «E» mit links getippt, also ist der obere Bogen für i-e, der untere für e-i.

Die grauen Balken unter der Tastatur zeigen die Anschlagshäufigkeit pro Finger. Der Kasten um den Balken entspricht 25 % der Anschläge, die Zahl darin ist die relative Häufigkeit bezogen auf die Zeichen. Der Kasten rechts in Abbildung 2c zeigt, dass der linke Zeigefinger etwa 14,3 % der Zeichen tippt. Dieser Wert ist etwas höher als in der Textausgabe, da dort der Grundwert die Zahl aller Anschläge, nicht, wie hier, die Zahl aller Zeichen ist.

Ganz unten sind Kästchen mit summierten Bigrammhäufigkeiten. Der Finger, für den ein Kästchen gezeigt wird, tippt das erste Zeichen des Bigramms. Kollisionen sind rot, die Intensitäten stehen für drei Bereiche der Sprungdistanz. Die Nachbaranschläge, in Ein- und Auswärtsbewegungen aufgeteilt, sind gelb-orange, die Farbintensität steigt mit den übersprungenen Zeilen (0, 1 oder 2). Die Kästchengrösse entspricht 2 % der Gesamtbigrammzahl, die Zahl im Kästchen ist die relative Häufigkeit bezogen auf die Gesamtzeichenzahl. In Abbildung 2c links unten sieht man, dass links der kleine Finger die meisten Kollisionen bewältigen muss, nämlich etwa 0,24 auf 100 Zeichen.

Jede Tastenzeile jeder Hand bildet eine *Gruppe*. Die grauen Balken links und rechts neben der Tastatur zeigen die Anschlagshäufigkeit pro Gruppe. Der Kasten um den Balken entspricht 40 % der Anschläge, die Zahlen darin sind auf die Zahl der Zeichen bezogen. Wie im mittleren grauen Kasten in Abbildung 2d zu sehen ist, entfallen etwa 37,9 % aller Zeichen auf die mittlere Zeile der

linken Hand.

Die Kästchen seitlich zeigen die Bigrammhäufigkeiten ohne Handwechsel, aufgeteilt nach der Gruppe, in die der erste Anschlag fällt. Für jede Gruppe wird weiter aufgeteilt nach der Zeile, in die der zweite Anschlag fällt. Die Kästchengrösse entspricht 20% der Gesamtbigrammzahl, die Zahl im Kästchen ist die relative Häufigkeit bezogen auf die Gesamtzeichenzahl. Die Zahl im grünen Kästchen unten in der Mitte von Abbildung 2d besagt also, dass auf 100 Zeichen etwa 6,7 Bigramme kommen, deren beide Anschläge in der mittleren Zeile der linken Tastaturhälfte liegen.

Wenn während einer Optimierung Grafiken erzeugt werden, sollte man mit Option `-i` die Zahl der Durchläufe festlegen. Bricht man die Optimierung manuell (mit Ctrl+C) ab, kann die Grafik unvollständig sein. Das PostScriptfile, das der Optimierer erzeugt, ist für Menschen les- und veränderbar. Am Anfang gibt es einige Schalter, die die Anzeige beeinflussen. Am Ende befindet sich die Liste der Belegungen, die dargestellt werden sollen.

5.3 Tippvorgang im Text markieren

Mit Option `-M` kann man den Namen eines UTF-8-codierten Textfiles angeben, für das für jede der Belegungen im mit Option `-r` angegebenen Belegungsfile gezeigt wird, wie die Eingabe erfolgt: welche Hand verwendet wird, ob das Zeichen in der Grundposition liegt, ob es ein Nachbaranschlag oder eine Kollision mit dem vorherigen Zeichen ist. Das Ergebnis steht in einem HTML-File, dessen Namen durch Anhängen von `.html` aus dem Namen des Textfiles entsteht. Am Anfang dieses Files befinden sich Definitionen, mit dem man den Stil der Darstellung einstellen kann.

6 Die Konfiguration ändern

Normalerweise holt der Optimierer seine Einstellungen aus `standard.cfg`, einem *Konfigurationsfile*, das beim Optimierer dabei ist. Die Option `-K` erlaubt es, ein anderes Konfigurationsfile anzugeben. Die Option darf mehrfach verwendet werden, was denselben Effekt hat, als würde man die so angegebenen Konfigurationsfiles aneinanderhängen.

Konfigurationsfiles sind UTF-8-codierte Textfiles mit zeilenorientierter Struktur. Leerzeilen werden ignoriert. Am Anfang jeder Zeile dürfen beliebig viele Leerzeichen und Tabulatoren stehen, die ignoriert werden. Ist das erste darauf folgende Zeichen ein #, wird der Rest der Zeile ignoriert. Andernfalls kommt ein Schlüsselwort, gefolgt von einer vom Schlüsselwort abhängigen Zahl von Argumenten, getrennt durch Leerzeichen oder Tabulatoren. Hinter den Argumenten darf ein mit # eingeleiteter Kommentar stehen. Argumente können *Namen* (Zeichenfolgen, die weder Leerzeichen noch Tabulatoren enthalten), *Strings* (in Anführungszeichen gesetzte Zeichenfolgen, wobei die Anführungszeichen am Anfang und Ende übereinstimmen müssen, sonst aber beliebig sind), *Flags* (+ oder −) oder Zahlen sein. Zahlen dürfen Dezimalkomma oder Dezimalpunkt enthalten.

Im Folgenden erscheinen einige mathematische Symbole in Klammern, auf die in Abschnitt 7 Bezug genommen wird.

6.1 Den Zeichenumfang festlegen

In der Regel wird man bei der Optimierung nur die wichtigsten Zeichen für die Sprachen, für die man eine Belegung sucht, berücksichtigen. Zeichen werden meist als Paare angegeben. Das erste Zeichen liegt auf der Grundebene, das zweite auf der geshifteten Ebene derselben Taste. Gibt man mehr als zwei Zeichen an, werden die zusätzlichen als Aliase des zweiten behandelt. Man kann auch ein einzelnes Zeichen angeben, das allein auf einer Taste liegt. Zum Beispiel bedeutet

```
Zeichen 'aA'
Zeichen ' '
```

dass das kleine und grosse «A» zusammen auf eine Taste kommen und das Leerzeichen eine eigene Taste bekommt. Man kann Zeichen auch einer bestimmten Taste zuordnen. Zum Beispiel wird mit

```
FixesZeichen AD11 'ß'
```

das «ß» auf die Taste mit dem Namen `AD11` gelegt (näheres zur Festlegung von Tasten und ihren Namen siehe Abschnitt 6.2). Die Position des «ß» wird dann nicht mitoptimiert, sein Vorhandensein in der Belegung auf `AD11` kann aber beeinflussen, welchen Tasten andere Zeichen zugeordnet werden.

Die Zahl von `Zeichen` und `FixesZeichen` im Konfigurationsfile muss der Zahl der Tasten weniger zwei (Shifttasten) sein, siehe `-DTASTENZAHL` in Abschnitt 1. Dasselbe Zeichen darf nicht in verschiedenen `Zeichen`- oder `FixesZeichen`-Festlegungen vorkommen.

Mit `Zeichen` und `FixesZeichen` kann man optional einen PostScript-Glyphnamen angeben, der das Symbol benennt, mit dem diese Zeichen in Grafiken dargestellt wird. Zum Beispiel legt

```
FixesZeichen SPCE ' ' underscore
```

fest, dass Leerzeichen in den Grafiken durch Unterstriche dargestellt werden. Die Angabe von Glyphnamen kann für Zeichen ausserhalb von ISO-8859-1 nötig sein. Zu deren Anzeige muss zudem die verwendete Schriftart diese Zeichen unterstützen. Die Schriftart lässt sich durch Angabe von PostScript-Fontnamen mit `Zeichenfont` und `Beschreibungsfont` für die Zeichen der Belegung und die Beschreibung (die Zahlen) getrennt angeben, zum Beispiel:

```
Zeichenfont       FiraMono-Medium
Beschreibungsfont FiraSans-Book
```

Manchmal will man die Grundebene zu einem Zeichen in der Shift-Ebene freilassen. Das erreicht man, indem man ein Platzhalterzeichen festlegt und damit die Grundebene belegt. Zum Beispiel kommt mit

```
Platzhalter '¦'
Zeichen '¦ß'
```

das «ß» auf die geshiftete Ebene einer Taste, deren Grundebene leer ist. Man kann nur einen Platzhalter festlegen, vor allen Zeichen. In der Auswertung wird der Platzhalter behandelt als käme er im Korpus nicht vor.

Man kann auch einzelne Zeichen durch eine Folge anderer ersetzen. Wenn man zum Beispiel statt «ß» lieber «ss» tippt, kann man das im Konfigurationsfile mit

```
Ersatz 'ßss'
```

ausdrücken. Das erste Zeichen in einem `Ersatz`-String ist das zu ersetzende Zeichen, der Rest der Ersatz. Ersetzt wird jedoch nur, wenn das zu ersetzende Zeichen nicht unter den mit `Zeichen` und `FixesZeichen` angegeben ist. Der Ersatz darf auch Zeichen enthalten, für die mit einem vorher angegebenen `Ersatz`-String ein Ersatz festgelegt wurde. Besonders interessant ist `Ersatz`, um tote-Taste-Folgen bei der Optimierung zu berücksichtigen.[2]

[2] `totetasten.cfg` zeigt ein Beispiel.

6.2 Das Tastaturlayout festlegen

Die Zahl der im Konfigurationsfile festgelegten Tasten muss der beim Übersetzen bestimmten entsprechen, siehe `-DTASTENZAHL` in Abschnitt 1. Ein Beispiel für die Festlegung einer Taste ist

```
Taste AD03   4 1   3.25 1   -3   -   4   -
```

Hier ist `AD03` der Name der Taste.[3] Tastennamen dürfen nur einmal vergeben werden.

Auf den Tastennamen folgen die Spalte und die Zeile (ξ und ρ) der Taste, im Beispiel `4` und `1`. Beides sind ganze Zahlen, Zeilen zwischen 0 und 4 (wobei 0 oben ist) und Spalten zwischen 0 und 15 (0 ist links). Dann kommen horizontale und vertikale Koordinaten (x und y) der Taste, im Beispiel `3.25` und `1`. Das sind beliebige Zahlen, die in der Regel jedoch nahe bei den Werten für Spalte und Zeile liegen.

Danach kommt eine Zahl, die den Finger (f) bezeichnet, der die Taste anschlägt. Negative Zahlen sind links, positive rechts. `-5` und `5` sind die kleinen Finger, `-4` und `4` die Ringfinger, `-3` und `3` die Mittelfinger, `-2` und `2` die Zeigefinger und `-1` und `1` die Daumen. `0` ist ein Daumen, wobei nicht festgelegt ist, welcher.

Dann kommt ein Flag. Ist es `+`, ist die Taste die Grundposition des entsprechenden Fingers. Die folgende Zahl gibt den Einzeltastenaufwand (α) der Taste an.

Schliesslich kommt optional noch ein Flag, das `+` oder `-` ist, wenn für diese Taste zur Ebenenumschaltung die linke oder rechte Shifttaste benutzt wird. Fehlt das Flag, wird für Tasten auf einem linken Finger die recht Shifttaste genommen und umgekehrt.

Für die Festlegung der linken und rechten Shifttaste gibt es `ShiftL` und `ShiftR`, die dieselben Argumente wie `Taste` erwarten. Die besonderen Schlüsselwörter werden verwendet, um sicherzustellen, dass beide Shifttasten tatsächlich im Konfigurationsfile festgelegt werden.

6.3 Die Bewertung festlegen

Einzelne Tasten

Die Einzeltastenaufwände werden mit den Tasten festgelegt, siehe Abschnitt 6.2. Die Zielhäufigkeit wird mit `Zielhäufigkeit` angegeben. `Zielhäufigkeit` hat zehn Argumente ($\tilde{\zeta}_{-5} \ldots \tilde{\zeta}_{-1}, \tilde{\zeta}_1 \ldots \tilde{\zeta}_5$), die unnormierten Zielhäufigkeiten für die Finger, beginnend mit dem linken kleinen Finger über linken und rechten Daumen bis zum rechten kleinen Finger. Mit `Fingerbelastung` werden entsprechend die zehn Gewichtsfaktoren ($\tilde{\phi}_{-5} \ldots \tilde{\phi}_{-1}, \tilde{\phi}_1 \ldots \tilde{\phi}_5$) angegeben, die bei der Überschreitung der Zielhäufigkeit benutzt werden, um eine Aufwand daraus zu bestimmen.

Bigramme

Für Bigramme gibt viele Parameter. Am allgemeinsten ist `Bigramm`, das zwei Tastennamen (in der Tippreihenfolge) und eine Zahl für den Aufwand ($\beta^{\mathrm{ex}}_{tt'}$) erwartet. Dieser Aufwand wird zu dem mit den anderen Parametern ermittelten addiert.[4] Beispiel:

```
Bigramm AD03 AB03 -2,5
```

Wahlweise kann man am Ende einer `Bigramm`-Zeile einen String angeben. Bigramme mit demselben String gehören zum selben Bigrammtyp, und ihre summierte Häufigkeit erscheint in der Textausgabe. Beispiel:

[3] Die Tastennamen in den mitgelieferten Konfigurationsfiles sind von xkeyboard-config übernommen und lehnen sich an ISO 9995-1 an. Wer will kann sie durch eingängigere Namen ersetzen, zum Beispiel `AD01` durch `Q`.

[4] Bigramme einzeln anzugeben wird schnell unhandlich. Das mitgelieferte `gencfg.awk` zeigt, wie man sich mit einem einfachen Skript behelfen kann.

```
Bigramm AC04 AC07 0 'Symmetrisch'
```

Dasselbe Bigramm kann in mehreren `Bigramm`-Zeilen vorkommen und somit zu verschiedenen selbstbestimmten Bigrammtypen gehören.

Die anderen Parameter für Bigramme drücken Aufwände in den Begriffen aus, die im Abschnitt 3 und 5 eingeführt wurden. Die Bigrammenklassen überlappen teilweise, zum Beispiel ist jede Auswärtsbewegung auch eine Handwiederholung. Die Aufwände, die für die verschiedenen Klassen veranschlagt werden, werden addiert.

`Handwechsel`, `Handwiederholung` und `Auswärts` erwarten jeweils eine Zahl, den Aufwand für einen Handwechsel (ω_1), eine Handwiederholung (ω_0), und eine Auswärtsbewegung ($\omega_{\leftrightarrow}$; ein Bigramm mit Daumenbeteiligung zählt nie als solche).

Mit `DoppeltRabatt` wird ein Aufwand für Doppeltanschläge berechnet, indem das Argument (ϱ_0) mit der Differenz aus dem Einzeltastenaufwand der Taste in der Grundposition des zugehörigen Fingers und dem Einzeltastenaufwand der Taste selbst multipliziert wird. Diese Differenz ist normalerweise negativ, daher bekommt man mit `DoppeltRabatt` üblicherweise eine Aufwandsreduktion. Von der Idee her ähnlich ist `ZeilenwiederholungRabatt`, das fünf Zahlen ($\varrho_1 \ldots \varrho_5$) erwartet. Wenn beide Anschläge mit verschiedenen Fingern (ohne Daumen) derselben Hand erfolgen und auf derselben Zeile liegen, diese nicht die mittlere Zeile (2) ist, die erste Taste im Bigramm nicht sowohl in der unteren Zeile liegt als auch mit dem kleinen Finger bedient wird, dann wird für die zweite Taste im Bigramm eine Differenz wie oben ermittelt und mit einem der Argumente von `ZeilenwiederholungRabatt` multipliziert. Das Argument wird gemäss der Spaltendifferenz der Tasten gewählt: Differenz 1 bedeutet erstes Argument, Differenz 2 zweites und so weiter. Zum Beispiel ist mit

```
ZeilenwiederholungRabatt 0.5 0.25 0.16666667 0.125 0.1
```

die Aufwandsreduktion bei Zeilenwiederholung umgekehrt proportional zur Spaltendifferenz.

Mit `KollisionKonstant` und `KollisionDistanz` kann man die Aufwände für Kollisionen festlegen. Beide Schlüsselwörter haben fünf Argumente ($\kappa_1 \ldots \kappa_5$ beziehungsweise $\varkappa_1 \ldots \varkappa_5$), eins pro Finger, beginnend mit dem Daumen auswärts. Der Aufwand ist der Wert aus `KollisionKonstant` plus das Produkt aus dem Wert aus `KollisionDistanz` und der Distanz, die der Finger bei der Kollision springen muss. Die Distanz ergibt sich aus den Koordinaten (x, y) der Tasten, siehe Abschnitt 6.2. Zum Beispiel legt

```
KollisionKonstant  10 10 10 10 10
KollisionDistanz   10 10 10 10 10
```

unter anderem fest, dass der Aufwand für Kollisionen unabhängig vom Finger ist.

Mit `Nachbar` werden die Aufwände für Nachbaranschläge festgelegt. Es gibt vier Argumente ($\nu_{3/2}, \nu_{5/2}, \nu_{7/2}, \nu_{9/2}$), für die Paarungen Daumen/Zeigefinger, Zeige/Mittelfinger, Mittel/Ringfinger und Ringfinger/Kleinfinger. Zum Beispiel werden mit

```
Nachbar  0 1.3333333 2 4
```

Nachbaranschläge als desto aufwändiger bewertet, je weiter aussen die beteiligten Finger liegen.

Für Handwiederholungen ohne Daumenbeteiligung gibt es zudem Aufwände für schräge Griffe, die mit `SchrägZS` und `SchrägYX` angegeben werden. Damit werden je zwei Zahlen angegeben (θ_{-1} und θ_1 beziehungsweise ϑ_{-1} und ϑ_1), die für die linke und rechte Hand gelten. Der Wert aus `SchrägZS` wird mit dem Verhältnis von Zeilendifferenz und Spaltendifferenz, der aus `SchrägYX` mit der Verhältnis von vertikalem zu horizontalem Abstand der Tasten multipliziert, um einen Aufwand zu erhalten. Zum Beispiel bewirkt

```
SchrägZS         0   1
SchrägYX         1   0
SchrägNenner0 0.1 0.1
```

dass schräge Griffe mit der linken Hand aufgrund der Koordinaten, solche mit der rechten Hand aufgrund der Spalten und Zeilen der Tasten berechnet werden. Mit `SchrägNenner0` werden hier zudem zwei Zahlen (Δ_{-1} und Δ_1, für die linke und rechte Hand) angegeben, die in dieser Berechnung sowohl zur Spaltendifferenz als auch zum horizontalen Abstand addiert werden.

Shift-Bigramme

Der Aufwand für ein Shift-Bigramm ergibt sich aus dem Aufwand des zugrunde liegenden Tastenbigramms durch Multiplikation mit einem der beiden Argumente (ς_+ und ς_-) von `Shiftbigramm`. Das erste wird benutzt, wenn der Aufwand des Tastenbigramms positiv ist, das zweite, wenn er negativ ist.

Trigramme

Die allgemeinste Festlegung von Trigrammaufwänden ist mit `Trigramm` möglich. Das funktioniert wie `Bigramm`, nur mit drei statt zwei Tastennamen.

Für Trigramme, deren erste und dritte Tasten mit derselben Hand angeschlagen werden, gibt es zusätzliche Beiträge. Zunächst wird der Aufwand für das Tastenbigramm aus erster und dritter Taste ermittelt. Er wird mit einem der beiden Werte (τ_+ und τ_-) multipliziert, die mit `Indirekt` angegeben werden: dem ersten, wenn der Aufwand positiv und dem zweiten, wenn er negativ ist. Wenn die mittlere Taste mit einer anderen Hand getippt wird, wird das Produkt als Zusatzaufwand genommen und dazu der mit `Doppelwechsel` angegebene Wert (ω_{11}) addiert. Wenn alle drei Tasten mit derselben Hand getippt werden, wird das Produkt mit der Summe der Bigrammaufwände von erster/zweiter und zweiter/dritter Taste verglichen; ist das Produkt grösser, ist der Zusatzaufwand das Produkt weniger der Summe, plus dem mit `Doppelwiederholung` angegeben Wert (ω_{00}). Sonst ist ist der Zusatzaufwand der mit `Doppelwiederholung` angegebene Wert.

Schliesslich wird für Trigramme, die aus einer Ein- und einer Auswärtsbewegung bestehen, der mit `Wippe` angegebene Wert ($\omega_{\rightleftarrows}$) zum Aufwand addiert.

Ähnlichkeit und Verwechslungspotenzial

Die allgemeinste Festlegung des Verwechslungspotenzials geht mit `Verwechslungspotenzial`. Das funktioniert wie `Bigramm`. Zusätzlich kann man Beiträge zum Verwechslungspotenzial von Tasten, die auf demselben (`VPKollision`, η) oder benachbarten Fingern (`VPNachbar`, υ) liegen angeben. `VPSymmetrisch` legt Beiträge (Σ) für Tasten fest, die mit denselben Finger an verschiedenen Händen bedient werden. Mit `VPSymmetrischGleicheZeile` gibt man Beiträge ($\Sigma_=$) für Tasten an, die zusätzlich noch in derselben Zeile liegen. `VPHandwechsel` (ϖ) ist für Tasten, die mit verschiedenen Händen bedient werden.

Mit `Ähnlich` wird die Ähnlichkeit (ε) zweier oder mehrerer Zeichen festlegt. Zum Beispiel kann man mit

```
Ähnlich 'bp' 0.1
Ähnlich 'sz' 0.05
```

sagen, dass sich «b» und «p» doppelt so ähnlich sind wie «s» und «z».

Vorlieben

Mit kann man bevorzugte Zuordnungen von Zeichen zu Tasten angeben. `Vorliebe` erwartet einen String mit Zeichen, die man derselben Tastenmenge zuordnen will, eine Zahl (λ), die die Stärke des Wunsches angibt, und die Menge Tasten, als Liste von Tastennamen. Um zu sagen, dass man x, c und v gerne auf den vier Tasten links unten hätte, kann man zum Beispiel angeben:

```
Vorliebe 'xcv' 0.1  AB01 AB02 AB03 AB04
```

Fehlende Zeichen

Mit `Fehlt` kann man einen Aufwand (Φ) für die Eingabe von Zeichen angeben, die in der Belegung nicht vorkommen. Zum Beispiel:

```
Fehlt 100
```

bewirkt, für dass jedes Zeichen, das im Korpus vorkommt und in der Belegung fehlt, ein Aufwand von 100 veranschlagt wird.

7 Die Aufwandsberechnung in Formeln

Wir gruppieren die Zeichen in der Belegung $\mathbb{B}$ in n Paare, ein Zeichen für die Grundebene und eins für die geshiftete Ebene. Jedes Zeichen $z \in \mathbb{B}$ ist durch ein Indexpaar $z = (p_z, e_z)$, mit $p_z \in \{0 \dots n-1\}$ und $e_z \in \{0, 1\}$, dargestellt. Für die n Zeichenpaare brauchen wir n Tasten, dazu zwei Shifttasten zur Ebenenwahl. Die Zeichentasten sind mit Null beginnend nummeriert, $t \in \{0 \dots n-1\}$. Die Shifttasten haben die Nummern n und $n+1$.

Eine Tasteneingabe i ist die Kombination von Zeichentaste t_i und Ebenenwahl e_i, dargestellt durch das Indexpaar $i = (t_i, e_i)$. Im Folgenden sind i, j und k solche Indexpaare, Summen über diese Indices laufen über alle Zeichentasten und Ebenen. t_i ist die erste Komponente (also die Taste) im Indexpaar i und e_i die zweite (die Ebene). Schliesslich ist s_i die zur Taste t_i gehörende Shifttaste.

Eine Belegung π ist eine Permutation von $(0 \dots n-1)$. Sie ordnet einer Tasteneingabe i ein Zeichen $z = \pi_i = (\pi_{t_i}, e_i)$ zu. Zeichen werden so immer paarweise Tasten zugeordnet. Die Ebene, auf der ein bestimmtes Zeichen liegt, bleibt fest.

Im Folgenden ist δ das Kronecker-Delta, $\bar{\delta}_{ij} = 1 - \delta_{ij}$ sein Komplement, Θ die Stufenfunktion

$$\Theta(x) = \begin{cases} 1 & \text{wenn } x > 0 \\ 0 & \text{sonst} \end{cases},$$

$\bar{\Theta}(x) = \Theta(-x)$ und $(x)_\pm = \pm x \Theta(\pm x)$ der Positiv- beziehungsweise der Negativteil von x.

7.1 Häufigkeiten

Die zu tippenden Texte sind durch Korpora gegeben, die wir mit einem Index $c \in K$ nummerieren. Korpus c hat ein Gewicht g_c, das sich durch Normierung aus den auf der Kommandozeile (mit `-G`) angegeben Gewichten ergibt, so dass $\sum_c g_c = 1$. Ein Korpus ist durch Tabellen mit Zeichenhäufigkeiten $H_i^{(c)}$, Bigrammhäufigkeiten $H_{ij}^{(c)}$ und Trigrammhäufigkeiten $H_{ijk}^{(c)}$ gegeben.

Sei $\mathbb{F}$ die Menge aller nicht-fixen Finger, f_t der Finger, der t bedient. Die Zahl von mit nicht-fixen Fingern getippter Zeichen in c ist $N^{(c)} = \sum_{f' \in \mathbb{F}} \sum_i \delta_{f' f_{t_i}} H_i^{(c)}$. Die Zahl der Anschläge mit

nicht-fixen Fingern für c ist $T^{(c)} = \sum_{f'\in\mathbb{F}}\sum_i(\delta_{f'f_{t_i}} + e_i\delta_{f'f_{s_i}})H_i^{(c)}$. Die relativen, gewichteten Gesamthäufigkeiten sind

$$h_i = \sum_{c\in K} g_c H_i^{(c)}/N^{(c)}\,, \tag{1a}$$

$$h_{ij} = \sum_{c\in K} g_c H_{ij}^{(c)}/N^{(c)} \quad \text{und} \tag{1b}$$

$$h_{ijk} = \sum_{c\in K} g_c H_{ijk}^{(c)}/N^{(c)}\,. \tag{1c}$$

Die relative Häufigkeit aller Zeichen, die im Korpus, aber nicht in der Belegung sind, ist

$$\bar{h} = \frac{\sum_{c\in K}\sum_{z\notin\mathbb{B}} g_c H_z^{(c)}}{\sum_{c\in K}\sum_{i\in\mathbb{B}} g_c H_i^{(c)}}\,.$$

7.2 Aufwände

Der gesamte Aufwand ist die Summe mehrerer Beiträge,

$$A(\pi) = A_0(\pi) + A_1(\pi) + A_2(\pi) + A_3(\pi) + A_\mathrm{f}(\pi) + A_\mathrm{fehlt}\,. \tag{2}$$

In der Textausgabe wird $100A$ als «Gesamtaufwand» angezeigt.

Der Beitrag zum Gesamtaufwand von Einzeltasten ist

$$A_1(\pi) = \sum_i a_i h_{\pi_i}\,, \tag{3}$$

wobei $a_i = \alpha_{t_i} + e_i(\alpha_{s_i} + \beta_{s_it_i})$ und die α die im Konfigurationsfile angegebenen Aufwände pro Taste sind. In der Textausgabe wird $100A_1$ als «Lageaufwand» angezeigt.

Die Beiträge zum Gesamtaufwand von den Zeichenbigrammen und Zeichentrigrammen sind

$$A_2(\pi) = \sum_{ij} a_{ij} h_{\pi_i\pi_j}\,, \tag{4}$$

$$A_3(\pi) = \sum_{ijk} a_{ijk} h_{\pi_i\pi_j\pi_k}\,, \tag{5}$$

mit

$$a_{ij} = (1-e_j)\Big[\beta_{t_it_j} + e_i\varsigma(\beta_{s_it_j})\Big] + e_j\Big[\beta_{t_is_j} + \gamma_{t_is_jt_j} + e_i\varsigma(\beta_{s_is_j})\Big]\,, \tag{6}$$

$$a_{ijk} = (1-e_j)\Big[(1-e_k)\gamma_{t_it_jt_k} + e_k\gamma_{t_it_js_k}\Big]\,, \tag{7}$$

siehe auch Tabelle 1 und 2. Die β und γ sind die Aufwände für Tastenbigramme und Tastentrigramme,

$$\begin{aligned}
\beta_{tt'} = {}& \beta^\mathrm{ex}_{tt'} + \delta_{1|f_t-f_{t'}|}\nu_{|f_t+f_{t'}|/2} + \delta_{tt'}\varrho_0(\alpha_{\Gamma_t} - \alpha_t) + \\
&+ \Theta(f_tf_{t'})\,\bar\delta_{tt'}\Big[\omega_0 + \Theta(|f_{t'}| - |f_t|)\,\bar\delta_{1|f_t|}\omega_{\leftrightarrow}\Big] + \bar\Theta(f_tf_{t'})\,\omega_1 + \\
&+ \Theta(f_tf_{t'})\,\bar\delta_{f_tf_{t'}}\bar\delta_{1|f_t|}\bar\delta_{1|f_{t'}|}\delta_{\rho_t\rho_{t'}}\bar\delta_{\rho_t 2}\left(1 - \delta_{3\rho_t}\delta_{5|f_t|}\right)\varrho_{|\xi_t-\xi_{t'}|}(\alpha_{\Gamma_{t'}} - \alpha_{t'}) + \\
&+ \lim_{\delta\to\Delta_{\chi_t}}\Theta(f_tf_{t'})\,\bar\delta_{f_tf_{t'}}\bar\delta_{1|f_t|}\bar\delta_{1|f_{t'}|}\left(\frac{\theta_{\chi_t}|\rho_t-\rho_{t'}|}{|\xi_t-\xi_{t'}|+\delta} + \frac{\vartheta_{\chi_t}|\gamma_t-\gamma_{t'}|}{|x_t-x_{t'}|+\delta}\right) + \\
&+ \bar\delta_{tt'}\delta_{f_tf_{t'}}\left(\kappa_{|f_t|} + \varkappa_{|f_t|}\sqrt{(x_t-x_{t'})^2 + (y_t-y_{t'})^2}\right)
\end{aligned}$$

und

$$\gamma_{tt't''} = \gamma^{\mathrm{ex}}_{tt't''} + \Theta\left(f_t f_{t''}\right)\left\{\bar{\Theta}\left(f_t f_{t'}\right)\left(\omega_{11} + \beta^{\mathrm{ind}}_{tt''}\right) + \right.$$
$$\left. + \Theta\left(f_t f_{t'}\right)\left[\omega_{00} + \Theta\left(\beta^{\mathrm{ind}}_{tt''}\right)\left(\beta^{\mathrm{ind}}_{tt''} - \beta_{tt'} - \beta_{t't''}\right)_+ + \Theta\Big(\left(f_t - f_{t'}\right)\left(f_{t''} - f_{t'}\right)\Big)\omega_{\rightleftarrows}\right]\right\},$$

wobei $\chi_t = \Theta\left(f_t\right) - \bar{\Theta}\left(f_t\right)$, $\beta^{\mathrm{ind}}_{tt'} = \tau_+\left(\beta_{tt'}\right)_+ - \tau_-\left(\beta_{tt'}\right)_-$ und $\varsigma\left(x\right) = \varsigma_+\left(x\right)_+ - \varsigma_-\left(x\right)_-$ ist. Γ_t ist die Taste in der Grundposition von f_t. Tasten mit gleicher Zeile und Spalte gelten als gleich, $\delta_{tt'} = \delta_{\rho_t\rho_{t'}}\delta_{\xi_t\xi_{t'}}$.

Ist $\zeta_f = \tilde{\zeta}_f / \sum_{f'}\tilde{\zeta}_{f'}$ die normierte Zielhäufigkeit für Finger f, die sich aus den $\tilde{\zeta}_f$ des Konfigurationsfiles ergibt, ist der Beitrag der Fingerbelastung zum Gesamtaufwand

$$A_{\mathrm{f}}\left(\pi\right) = \sum_{c\in K} g_c \sum_{f\in\mathbb{F}} \phi_f\Big(b^{(c)}_f\left(\pi\right) - \zeta_f\Big)^2_+ , \tag{8}$$

wobei $b^{(c)}_{f'}\left(\pi\right) = \sum_i (\delta_{f'f_{t_i}} + e_i\delta_{f'f_{s_i}})H^{(c)}_{\pi_i}/T^{(c)}$ die Belastungen des Finger f' für Korpus c ist und $\phi_f = \tilde{\phi}_f$ falls ohne und $\phi_f = (1+\tau_+)\,\tilde{\phi}_f$ falls mit Trigrammen bewertet wird.

Die nichtmechanischen Kriterien setzen sich aus Verwechselbarkeit und Vorlieben zusammen,

$$A_0\left(\pi\right) = \sum_{t=0}^{n-1}\sum_{t'=0}^{n-1}\varphi_{tt'}\varepsilon_{\pi_t\pi_{t'}} - \sum_{t=0}^{n-1}\lambda_{\pi_t t}\,. \tag{9}$$

φ ist das Verwechslungspotenzial zweier Tasten,

$$\varphi_{tt'} = \varphi^{\mathrm{ex}}_{tt'} + \bar{\delta}_{tt'}\delta_{f_t f_{t'}}\eta + \delta_{1|f_t - f_{t'}|}\upsilon + \left[1 - \Theta\left(f_t f_{t'}\right)\right]\varpi +$$
$$+ \bar{\Theta}\left(f_t f_{t'}\right)\bar{\delta}_{1|f_t|}\bar{\delta}_{1|f_{t'}|}\delta_{|f_t||f_{t'}|}\left(\bar{\delta}_{\rho_t\rho_{t'}}\Sigma + \delta_{\rho_t\rho_{t'}}\Sigma_=\right),$$

ε die Ähnlichkeit zweier Zeichen und λ die Vorliebe dafür, ein bestimmtes Zeichenpaar auf eine bestimmte Taste zu legen.

Der Aufwand für in der Belegung fehlende Zeichen ist $A_{\mathrm{fehlt}} = \bar{h}\Phi$.

7.3 Pareto-Eigenschaft

Dass die Belegung, die A minimiert, Pareto-optimal bezüglich der einzelnen Korpora ist, liegt daran, dass die Aufwände $A^{(c)}$ der einzelnen Korpora sich zum Gesamtaufwand summieren,

$$A = \sum_{c\in K} g_c A^{(c)}\,. \tag{10}$$

Wäre das Optimum π_0 nicht Pareto-optimal, gäbe es eine Belegung π_1 und ein $c_0 \in K$ mit $A^{(c_0)}\left(\pi_1\right) < A^{(c_0)}\left(\pi_0\right)$ und $A^{(c)}\left(\pi_1\right) \leq A^{(c)}\left(\pi_0\right)$ für $c \neq c_0$. Damit würde jedoch die Summe kleiner, was der Annahme widerspricht, dass π_0 optimal ist.

Die Summeneigenschaft (10) für A_{f} ist explizit in Gleichung (8). A_1, A_2 und A_3 (Gleichung (3), (4) und (5)) sind linear in den h, daher kann man die c-Summation aus der Definition (1) der h herausziehen, zum Beispiel $A_1\left(\pi\right) = \sum_i a_i \sum_c g_c H^{(c)}_{\pi_i}/N^{(c)} = \sum_c g_c \sum_i a_i H^{(c)}_{\pi_i}/N^{(c)} = \sum_c g_c A^{(c)}_1\left(\pi\right)$. A_0 in Gleichung (9) schliesslich ist korpusunabhängig, somit $A_0\left(\pi\right) = A^{(c)}_0\left(\pi\right)$, und die Summeneigenschaft gilt wegen $\sum_c g_c = 1$.

Verwendet man hingegen ein einzelnes Korpus, den man extern aus verschieden Teilkorpora zusammengesetzt hat, verhindert die Nichtlinearität von Gleichung (8) in h, dass man garantieren kann, dass das Gesamtoptimum bezüglich der Teilkorpora Pareto-optimal ist.

7.4 Statistischer Fehler

Wir betrachten zunächst ein einziges Korpus. Wir stellen uns vor, dieser sei eine Stichprobe aus einem unendlich grossen, idealen Korpus und dadurch gebildet, dass wir wiederholt zufällig und voneinander unabhängig ein Wort aus dem idealen Korpus wählen. Diese Annahme ist die entscheidende Vereinfachung. In Wirklichkeit sind die unkorrelierten Einheiten wohl grössere als einzelne Wörter.

Die Wahrscheinlichkeit, bei einer Wortwahl das Wort w zu wählen, sei p_w. Die Häufigkeit, mit der das Wort w ausgewählt wird, ist binominalverteilt mit Mittelwert $\mu_w = p_w H_{\text{wort}}$ und Varianz $\sigma_w^2 = H_{\text{wort}} p_w (1 - p_w)$. Wir können diese Parameter aus den beobachteten Häufigkeiten als $\mu_w = H_w$ und $\sigma_w^2 = H_w (1 - H_w/H_{\text{wort}})$ schätzen, wobei H_w die Häufigkeit des Worts w und H_{wort} die Grösse der Stichprobe sind.

Die Häufigkeit, mit der ein Zeichen i und das Zeichenbigramm ij in w vorkommen, bezeichnen wir mit w_i und w_{ij}. Die Anzahl der Zeichen in einem Wort ist $|w| = \sum_i w_i$, die Anzahl der Anschläge, die man zur Eingabe benötigt, $\|w\| = \sum_i (1 + e_i)\, w_i$. Damit kann man die Zahl der Zeichen und der Anschläge in der Stichprobe als $N = \sum_w |w| H_w$ und $T = \sum_w \|w\| H_w$ schreiben, die relative Häufigkeit des Zeichens i als $h_i = \sum_w w_i H_w / N$ und so weiter.

Die Worthäufigkeiten in einer Stichprobe werden von ihrem Erwartungswert abweichen. Die Abweichung $\delta A(\pi^{-1})$ vom Erwartungswert des Aufwands $A(\pi^{-1})$ der Belegung π^{-1} aufgrund dieser Abweichungen δH_w ist

$$\delta A(\pi^{-1}) = \sum_w \left[\sum_i a_{\pi_i} \frac{w_i - |w|\, h_i}{N} + \sum_{ij} a_{\pi_i \pi_j} \frac{w_{ij} - |w|\, h_{ij}}{N} + \right.$$
$$\left. + \sum_{f' \in \mathbb{F}} 2 \phi_{f'} \left(b_{f'} - \zeta_{f'}\right)_+ \sum_i (\delta_{f' f_{t_{\pi_i}}} + e_i \delta_{f' f_{s_{\pi_i}}}) \frac{w_i - \|w\| \bar{h}_i}{T} \right] \delta H_w \,,$$

wobei $\bar{h}_i = H_i / T$ und die Fingerbelastung und die Gesamtzeichenzahl um die Erwartungswerte der Worthäufigkeiten linearisiert wurden. Da wir die Häufigkeiten verschiedener Wörter als unabhängig betrachten ist $\langle \delta H_w \delta H_{w'} \rangle = \sigma_w^2 \delta_{ww'}$, wobei $\langle \cdot \rangle$ der Erwartungswert ist. Damit wird

$$\left\langle \delta A^2(\pi^{-1}) \right\rangle = \sum_w \left[\sum_i a_{\pi_i} \frac{w_i - |w|\, h_i}{N} + \sum_{ij} a_{\pi_i \pi_j} \frac{w_{ij} - |w|\, h_{ij}}{N} + \right.$$
$$\left. + \sum_{f' \in \mathbb{F}} 2 \phi_{f'} \left(b_{f'} - \zeta_{f'}\right)_+ \sum_i (\delta_{f' f_{t_{\pi_i}}} + e_i \delta_{f' f_{s_{\pi_i}}}) \frac{w_i - \|w\| \bar{h}_i}{T} \right]^2 H_w \left(1 - \frac{H_w}{H_{\text{wort}}}\right) .$$

Zur Auswertung multipliziert man die eckigen Klammern aus. Aus den Produkten der Terme $(w_i - |w|\, h_i)/N$, $(w_{ij} - |w|\, h_{ij})/N$ und $\left(w_i - \|w\| \bar{h}_i\right)/T$ entstehen sechs Korrelationen, zum Beispiel

$$\sigma_{ij;k}^{(2\text{f})} = \sum_w \frac{w_{ij} - |w|\, h_{ij}}{N} \frac{w_k - \|w\| \bar{h}_k}{T} H_w \left(1 - \frac{H_w}{H_{\text{wort}}}\right) .$$

Man kann sie direkt aus dem Korpus gewinnen und daraus $\langle \delta A^2 \rangle$ leicht berechnen,

$$\left\langle \delta A^2(\pi^{-1}) \right\rangle = \ldots + 2 \sum_{ijk} a_{\pi_i \pi_j} \sum_{f' \in \mathbb{F}} \phi_{f'} \left(b_{f'} - \zeta_{f'}\right)_+ (\delta_{f' f_{t_{\pi_k}}} + e_i \delta_{f' f_{s_{\pi_k}}}) \sigma_{ij;k}^{(2\text{f})} + \ldots$$

Um die Varianz einer Aufwandsdifferenz zu berechnen, ersetzt man die π-abhängigen Vorfaktoren durch Differenzen für die beiden Belegungen π und π', zum Beispiel $a_{\pi_i \pi_j}$ durch $a_{\pi_i \pi_j} - a_{\pi'_i \pi'_j}$. Die Gesamtvarianz für mehrere Korpora ergibt sich aus den mit g_c^2 gewichtetet Einzelvarianzen.

8 Anhang

8.1 Die mitgelieferten Häufigkeitsfiles

Das deutsche Korpus, der den Files `deutsch.txt.*` zugrunde liegt, setzt sich so zusammen:

- 636 kB aus zufällig gewählten Zeilen des von Karl Köckemann nachbearbeiteten deutschsprachigen Leipziger Korpus. Das Orginalkorpus enthält 3 Millionen Sätze aus verschiedenen Zeitungen. Karls Bearbeitung passt ihn an die neue Rechtschreibung an.
- 642 kB aus den Essays, die in der Zeitschrift c't zwischen 22/1999 und 12/2006 publiziert wurden. Diese Essays benutzen die neue Rechtschreibung und liegen als HTML vor. Die HTML-Formatierung wurde entfernt, ebenso Überschriften und Paragrafen, die `<a>`, `<code>` und ähnliches enthalten.
- 150 kB Sachtexte und 200 kB literarische Texte von Project Gutenberg:
  Sigmund Freud, Massenpsychologie und Ich-Analyse, Kapitel I und II; Arnold Sommerfeld, Relativitätstheorie, aus Deutsches Leben in der Gegenwart; Mihai Nadin, Jenseits der Schriftkultur, 40 kB aus Buch I, Kapitel I; Alexander Lipschütz, Warum wir sterben (ca. 40 kB); Gerhart Hauptmann, Bahnwärter Thiel (ca. 20 kB vom Anfang); Stefan Zweig, Brennendes Geheimnis (ca. 20 kB vom Anfang); Robert Walser, Der Gehülfe (ca. 20 kB vom Anfang); Gottfried Keller, Die Leute von Seldwyla, Vol I (ca. 20 kB vom Anfang); Franz Kafka, Die Verwandlung (ca. 20 kB vom Anfang); Alfred Döblin, Die Ermordung einer Butterblume und andere Erzählungen (ca. 20 kB vom Anfang); Hermann Hesse, Siddhartha (ca. 20 kB vom Anfang); Thomas Mann, Der Tod in Venedig (ca. 20 kB vom Anfang); Rainer Maria Rilke, Zwei Prager Geschichten (ca. 20 kB vom Anfang).
  Diese Texte verwenden die alte Rechtschreibung, lediglich «daß» wurde durch «dass» ersetzt und Paragrafen zu Zeilen zusammengezogen.

Für die `deutsch-t.txt`-Files wurden die Trennmuster `hyph-de-1996.pat.txt` (Stand Mai 2014) und dasselbe Korpus verwendet.

Das englische Korpus, der den Files `englisch.txt.*` zugrunde liegt, setzt sich so zusammen:

- 920 kB aus zufällig gewählten Zeilen des englischen Leipziger Korpus. Amerikanische Schreibweisen wurden teilweise in englische umgewandelt (zum Beispiel «center» in «centre»). Einige Einheiten wurden durch international gültiges ersetzt (zum Beispiel «miles» durch «kilometres»).
- 417 kB Sachtexte:
  - Aus O'Reilley open books: Stephen L. Talbott, The future does not compute, Kapitel 23, gefiltert wie die c't-Essays.
  - Von Project Gutenberg:
    Charles Darwin, On the Origin of Species, Kapitel I; James Clerk Maxwell, Five of Maxwell's Papers.
  - Aus dem Open American National Corpus, Verzeichnis `written_2/non-fiction/OUP`:
    Abernathy, Kapitel 3; Berk, Kapitel 7; Fletcher, Kapitel 10; Kauffmann, Kapitel 5; Rybczynski, Kapitel 1.
- 339 kB Literatur von Project Gutenberg:
  Lewis Carroll, Alice in Wonderland; Joseph Conrad, Heart of Darkness, Kapitel; W. Somerset Maugham, Of Human Bondage, Kapitel 1–16; James Joyce, Ulysses, ca. 2500 Zeilen.

Für die `englisch-t.txt`-Files wurden die Trennmuster `hyph-en-gb.pat.txt` (Stand Mai 2010) und dasselbe Korpus verwendet.

8.2 Beispielbelegungen

Näheres zu den Tastaturbelegungen in `bsptast.txt`: Arensito, Aus der Neo-Welt, Colemak, Klausler, KOY, Neo 2. Zum Teil habe ich die Umlaute selbst ergänzt.

8.3 Weitere Informationen

Der Optimierer hat eine Homepage, über die die aktuelle Version und weitere Informationen verfügbar sind. Ein guter Startpunkt, um mehr über den Hintergrund, auf dem Beurteilungen von Belegungen beruhen, zu lernen, ist `http://adnw.de`.

1 Preparation: Compiling the program

Before you can use the optimiser, you must compile it. Compiling translates the human-readable source code into a format that is directly executable by the computer. To compile, you need a C++ compiler, for example the free GNU C++ compiler (version 4.8.1 or newer). With this compiler, compilation is done by:

```
g++ -std=c++11 -O2 -DNDEBUG -DENGLISH opt.cc -o opt
```

This results in the executable program `opt`. Using the another free choice, the LLVM C++ compiler (version 3.3 or newer), the the command for compilation is:

```
clang++ -stdlib=libstdc++ -std=c++11 -O2 -DNDEBUG -DENGLISH opt.cc -o opt
```

The optimiser can be compiled for a given number of keys. For example,

```
g++ -std=c++11 -O2 -DNDEBUG -DTASTENZAHL=32 -DENGLISH opt.cc -o opt
```

creates an executable for 32 keys. The default is 35 keys, which includes two shift keys. If you change the number of keys from its default, you have to change the configuration as well, see Section 6.

To reach the highest possible speed, consider experimenting with the compiler options. For example, try

```
g++ -std=c++11 -Wall -Ofast -DENGLISH -DNDEBUG -DOHNE2SHIFT \
    -DMIT_THREADS -pthread opt.cc -o opt
```

Above, the option `-DOHNE2SHIFT` causes upper case letters that occur as the second character in a sequence of two characters to be ignored. Upper case letters usually occur only at the beginning of words (that is, as the first character). Therefore, this simplification typically has little impact on the optimisation result.

If you want the optimiser to support multi-threading (that is, the concurrent use of multiple processor cores), you have to specify the option `-DMIT_THREADS` for compilation. On many systems, additional options are required. For example, on my system, I need the option `-pthread`.

By default, the optimiser uses UTF-8 character encoding. Using the option `-DAUSGABE_8BIT`, you can switch the encoding for terminal output to ISO-8859-1.

2 What to type: Corpus and frequency files

To use a computer to design a good keyboard layout, one has to specify what one plans to type. To this end, a *corpus* is used, that is, a collection of texts that are representative of those one wishes to type.

The optimiser only uses a part of the information in the corpus: The letter frequencies, the frequency of digrams (sequences of two letters) and of trigrams (sequences of three letters). These frequencies are stored in files which share the same name stem, and have extensions `.1`, `.2`, and `.3`, respectively. Digrams and trigrams do overlap. For example, the word “bike” contains the digrams “bi”, “ik”, and “ke”, as well as the trigrams “bik” and “ike”.

The optimiser ships with frequency files for German and English. For the files with `-t` in their name, only digrams and trigrams are accounted for that are in the interior of words and that do not contain hyphenation points.

2.1 Creating your own frequency files

To use your own collection of texts, proceed as follows:

1. Put all the texts in a single file, for example `meinkorpus.txt`.
2. Make sure that this file is UTF-8 encoded.
3. Use `opt` to create the frequency files. For example,

```
./opt meinkorpus.txt
```

   creates the frequency files `meinkorpus.txt.1`, `meinkorpus.txt.2`, and `meinkorpus.txt.3` from the text collection in file `meinkorpus.txt`, as well as a word list in `meinkorpus.txt.wl`.

If you invoke `opt` with two arguments, the optimiser assumes that the first argument denotes a file containing UTF-8 encoded TEX hyphenation patterns (`*.pat.txt`), and the second argument is the file with the text collection. The optimiser then creates frequency files that account only for digrams and trigrams that are in the interior of words and do not contain a hyphenation point. Instead of a word list, a list of syllables is created. For example:

```
./opt hyph-de-1996.pat.txt meinkorpus.txt
```

`opt` can also add frequency files created from different text collections. To that end, invoke `opt` with more than two arguments. For example,

```
./opt deutsch.txt deutsch-t.txt deutsch-t.txt gemischt.txt
```

uses the frequency files for German that are shipping with the optimiser to create new frequency files in which digrams and trigrams at word boundaries or containing hyphenation points are only accounted with one third of their weight. Another possibility for combining and weighting frequency files is option `-G`, as described below.

2.2 Using frequency files

To run the optimiser, it is provided with the name stem of the frequency files. To do so, use the options `-2` or `-3`. Using option `-2`, only letter and digram frequencies will be used. Using `-3`, additionally, trigram frequencies will be accounted for. For example,

```
./opt -2 deutsch.txt
```

runs an optimisation with the frequency files for German, not accounting for trigrams.

In case the frequency file for the letter frequencies (in the example above, `deutsch.txt.1`) does not exist, the optimiser tries to open a file with the name of the given argument (that is, `deutsch.txt`). If it succeeds, this file is treated as a corpus. This allows it to skip the intermediate step of using frequency files, however, reading the corpus takes more time.

You can specify `-2` or `-3` multiple times. By default, all frequency files specified that way have the same weight, irrespective of the size of the corpus used to create them. Using option `-G` (default 1), you can change the weight for frequency files specified subsequently, see Equation (1). For example, using

```
./opt -2 deutsch.txt -G 3 -2 englisch.txt
```

runs an optimisation for which English enters with three times the weight of German.

Specifying different frequency files separately by using multiple `-2` or `-3` options takes somewhat higher computational effort than using a single summed corpus. On the other hand, the advantage of doing so is that the optimum layout determined this way (assuming that it is found at all) is a Pareto optimum with respect to the individual corpora. That is, it is not possible to further

improve the layout for any of the corpora without making it worse for another one. Using just a single sum corpus does not guarantee this property (see Section 7.3).

2.3 Size of the corpus, systematic and statistical errors

If instead of using frequency files you read a full corpus directly, when evaluating layouts stored in a file (see option `-r` in Section 5), the standard deviation of the effort and the standard deviation of the difference to the effort of the first layout in the file are estimated and printed. For the estimation, we assume that the frequencies or different words in a text are uncorrelated. A "word" in this context is a sequence of characters, limited by a space or a character that is not part of the layout (see Section 6.1). For the frequencies of letters, we assume a binomial distribution, and the average and the variance of the distribution are estimated from the relative frequencies of words found in the corpus. For the computation of the standard deviation, trigrams and missing symbols are not taken into account.

Differences of efforts of a single standard deviation or below are not significant, that is, they can easily be caused by the random selection of texts alone. Differences of three standard deviations or more can be considered as real. To reduce the standard deviation, you must increase the size of the corpus. The standard deviation is inverse proportional to the square root of the corpus size. For example, to reduce the standard deviation by half, you need a corpus four times as large.

The statistical error is not the only uncertainty that stems from the corpus. For example, the type of texts (texts with long or short sentences, with many or with few foreign words) enters in a systematic way that is not visible in the standard deviation. From my experience, a corpus of a few megabytes is large enough.

2.4 Replacing characters by others

If you want to enter a single character by typing a sequence of one or several other characters, you can express this by using `Ersatz`, see Section 6.1. It is not necessary to modify the corpus or the frequency files.

However, if you want to replace a sequence of multiple characters by a sequence of characters, this simple procedure does not work anymore, as the information retained in the frequency files is not sufficient to support general replacements. In this case, you have to replace the character sequence to generate by the character sequence to type in the corpus directly. From this modified corpus, you then generate new frequency files.

2.5 Other operations

Using option `-T`, from a corpus and a file with hyphenation patterns, you can create another file in which all possible hyphenation points are marked by a soft hyphen (U+00AD). For example:

```
./opt -T hyph-en-gb.pat.txt input.txt result.txt
```

3 The evaluation scheme

To evaluate a keyboard layout, we consider the sequence of key strokes required to enter the corpus with this layout. Similar to the corpus, we restrict our evaluation to a couple of criteria. Each of these criteria contributes an *effort*. The sum of these contributions is the total effort, see Equation (2).

digram	keys	key di- and trigrams
xy	$k_x k_y$	$k_x k_y$
Xy	$s_x k_x k_y$	$s_x k_x k_y$, $k_x k_y$, $s_x k_x$
xY	$k_x s_y k_y$	$k_x s_y k_y$, $s_y k_y$, $k_x s_y$
XY	$s_x k_x s_y k_y$	$s_x k_x s_y$, $k_x s_y k_y$, $s_x k_x$, $k_x s_y$, $s_y k_y$

Table 1: Decomposition of digrams into key di- and trigrams. k_X is the key for x, s_X is the associated shift key, k_Y and s_Y analogeous. $s_X k_X$ and $s_Y k_Y$ are counted as individual keys, as they correspond to uppercase letters. $k_X k_Y$ and $k_X s_Y$ are is a normal key digrams, $s_X k_X k_Y$ and $s_X k_X s_Y$ are shift digrams, and $k_X s_Y k_Y$ a key trigram.

3.1 Mechanical criteria

Individual key efforts. Each key has an effort assigned to it. This effort is multiplied with the frequency with which the key is pressed, that is, with the frequency of the letter that is mapped to the key. The sum of these products is the *positional effort*, see Equation (3).

Digram efforts. Each sequence of two keys has an effort, the digram effort. Here, we call a sequence of two key strokes *digram*, just like we call a pair of characters. To avoid ambiguity, we will sometimes use the term *key digram*. The difference is show in table 1 by decomposing the digram "xy" with all combinations of upper and lower case charaters. This leads to key digrams of the form shift+character key, the frequencies of which are taken from the letter frequencies, and the effort of which are part of the positional effort. Furthermore, normal key digrams made up of two character keys, shift digrams and trigrams appear, so that the contribution from digrams gets quite complex, see Equation (6).

Shift digrams consist of one shift key, the associated character key, and the following key. For their effort, the digram effort for the shift key and the last key is multplied by a factor, see Equation (6).

Trigram efforts. A sequence of three key presses is given an effort, the trigram effort. Only when using option `-3`, all key trigrams are accounted for. Otherwise, only key trigrams the frequency of which is given by digram frequencies are accounted for, see table 2 and Equation (7).

Finger load. For each finger, a *target frequency* is specified, which denotes the fraction of the total number keystrokes the finger should type. If the actual frequency exceeds the target frequency, the surplus is squared and multiplied with a weight factor to give an effort, see Equation (8). *Fixed fingers* are an exception: Fingers the key stokes of which are not dependent on the layout, but are fully determined by the configuration and the corpus.

3.2 Non-mechanical evaluation criteria

The evaluation described up to now is concerned with the mechanics of typing. Possibly, the frequency with which similar letters are confused depends on the relative positioning of the keys

trigram	keys	key trigrams
xyz	$k_x k_y k_z$	$k_x k_y k_z$
Xyz	$s_x k_x k_y k_z$	$s_x k_x k_y$, $k_x k_y k_z$
xYz	$k_x s_y k_y k_z$	$k_x s_y k_y$, $s_y k_y k_z$
XYz	$s_x k_x s_y k_y k_z$	$s_x k_x s_y$, $k_x s_y k_y$, $s_y k_y k_z$
xyZ	$k_x k_y s_z k_z$	$k_x k_y s_z$, $k_y s_z k_z$
XyZ	$s_x k_x k_y s_z k_z$	$s_x k_x k_y$, $k_x k_y s_z$, $k_y s_z k_z$
xYZ	$k_x s_y k_y s_z k_z$	$k_x s_y k_y$, $s_y k_y s_z$, $k_y s_z k_z$
XYZ	$s_x k_x s_y k_y s_z k_z$	$s_x k_x s_y$, $k_x s_y k_y$, $s_y k_y s_z$, $k_y s_z k_z$

Table 2: Decomposition of trigrams into key trigrams. k_X is the key for x, s_X is the associated shift key, k_Y, s_Y, k_Z and s_Z analogeous. Only for the frequencies of $k_X k_Y k_Z$ und $k_X k_Y s_Z$, we need trigram frequencies. For the other key trigrams and shift digrams, digram frequencies are sufficient.

those letters are mapped to. In the configuration file, one can specify the degree of similarity of characters by a number. This number is multiplied with the *confusability* to obtain an effort, see Equation (9). The confusability depends on the two keys the letters are mapped to, and can be specified in the configuration file. By default, all letters are treated as dissimilar.

Some users want certain characters to be mapped to certain parts of the keyboard layout. *Preferences* for the mapping of characters to keys like this can be defined in the configuration file, with a number which quantifies the importance of this preference. Each fulfilled preference reduces the effort by this number, see Equation (9).

The manipulation of corpus and frequency files opens additional possibilities to affect the evaluation. For example, in a posting to the Neo mailing list, it was claimed that the flow of writing is more important within a syllable than across syllable boundaries. If for optimisation, one uses frequency files in which only di- and trigrams are counted that do not contain a hyphenation point (see Section 2), it is possible to account for this in an approximate manner (approximate, as syllable boundaries and hyphenation points are not exactly the same thing).

For comparison of layouts that support different sets of symbols, it is possible to specify an effort for symbols that cannot be entered with a layout.

3.3 Printing evaluation criteria

Using the option `-A`, you can print all non-zero digram efforts, confusabilities, and preferences. If you have specified `-3` as well, trigram efforts will be output, too.

4 How optimisation proceeds

The optimiser uses a simple algorithm that starts out from a random keyboard layout. This layout is improved incrementally by repeatedly swapping characters. If no swap improves the layout anymore, the algorithm terminates. The result of this procedure is a *locally optimal* keyboard layout.

A locally optimal keyboard layout may be far from optimal, in the sense given by the evaluation scheme. For this reason, the procedure outlined above is repeated over and over again, using different random layouts as starting point. The number of repetitions can be specified with option `-i`. Without this option, the optimiser keeps running until you terminate it. How many repetitions you should allow for depends on the evaluation scheme and, possibly, on the corpus. According to my observations, using the default settings, 10000 is a reasonable value, assuming you are using option `-2`. Using option `-3` tends to require more repetitions.

By default, each locally optimal keyboard layout that is better than any encountered before will be printed. If you use option `-m` to define a threshold, all locally optimal keyboard layouts with a total effort below this threshold will be printed.

Using option `-s`, you can specify a positive integer which serves as the seed for the random number generator. This can be useful to get reproducible optimiser runs. By default, the seed is picked randomly.

The number of threads to use can be specified with option `-t`. By default, one thread is used. The number of repetitions specified with `-i` is understood per thread. That is, if you use more threads, you can decrease this number in proportion.

5 The output of the optimiser

We use the following terms: *Hand alternations* are digrams the keys of which are typed by different hands. *Double strokes* are digrams for which the same key is struck twice. *Same finger repetitions* are digrams for which different keys are struck with the same finger. *Adjacent finger strokes* are digrams for which the keys are typed by adjacent fingers of the same hand. *Inward motions* are digrams for which the first key is typed by a finger further outward on the same hand than the finger typing the second key (the outermost finger is the pinkie, the innermost is the thumb). For *outward motions*, the order is opposite. The prefix "Shift-" denotes special digrams, for which the first key is shift and the second key is a letter key.

5.1 Text output

Here is the output for "Aus der Neo-Welt", evaluated with a mixed German-English corpus:

```
 Aus der Neo-Welt 382.859 total effort   187.075 positional effort    left right
                    1.029 same finger rp   6.976 shift same finger top  5.7 11.8
   kuü.ä vgcljf    71.404 hand alternat.  24.118 shift hand alter. mid 36.4 32.1
   hieao dtrnsß     1.796 inward/outward  25.117 inward or outward bot  5.2  8.9
   xyö,q bpwmz      9.262 adjacent        22.116 shift adjacent    sum 47.2 52.8
                   8.4 11.2 14.0 13.7 --.- --.- 17.6 10.8 14.3 10.1 Sh  2.9  1.2
```

On the top left, the name of the layout or a sequential number is printed. Next to it, the total effort and the positional effort are printed. To the left, the layout is shown. Next to it, digram frequencies are listed: The percentage of all key digrams that are same finger repetitions, hand alternation or adjacent finger strokes, and the ratio of inward and outward motions. The column right to this contains the percentages of shift digrams that are same finger repetitions, hand alternations or adjacent finger strokes. On the right side, the distribution of key strokes is detailed for upper, middle, and lower row, and the two hands. The line on the bottom shows how keystrokes distribute among fingers (starting from the left pinkie up to the right pinkie) and the shift keys. The `--.-` for the thumbs comes from the space having been assigned to a thumb, but not to a particular one. For this reason, it is not possible to specify frequencies for the individual thumbs. Key stokes for an unspecified thumb are accounted for in the total and the positional effort, but are ignored for the rest of the output.

When using option `-3`, two additional lines are printed:

```
 4.765 no hand altern. 44.851 two hand altern.
 3.582 seesaw           6.403 indir same finger
```

The first one gives the frequencies of trigrams with no and with two hand alternations, the second one gives the frequencies of trigrams that consist of one inward and one outward motion (no matter their order), and the frequency of trigrams with two hand alternations, for which the first and last key is different, but typed by the same finger (*indirect same finger repetition*).

Using option `-b`, followed by a number between zero and 100, you can get a more verbose description of the most frequent digrams without hand alternation. In addition, the frequency of same finger repetitions and adjacent finger strokes will be detailed per finger and per finger pair, respectively. Using option `-b` a second time (followed by a number) yields the shift digrams in addition, using it a third time yields in the trigrams in addition.

Using option `-k` reduces output to one line per keyboard layout. Option `-m` specifies a total effort below which all locally optimal layouts are printed. Without this option, only the currently best layout will be displayed.

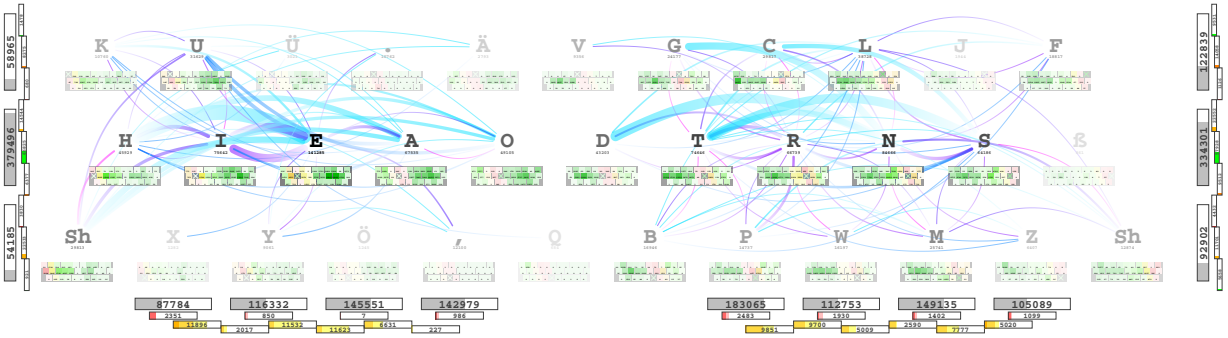


Figure 1: Example for the layout graphics

Using option `-w`, you can specify a file that contains a word list. The file must be UTF-8 encoded, and each line of it must contain a word frequency followed by the word, separated by a space. For a given keyboard layout, the words are split into letter sequences that are typed with the same hand. The frequencies of these *one-handed sequences* are collected and summed over all words, and then output sorted according to their frequency. By default, as many one-handed sequences will displayed so that their cumulative frequency is 95 % of all one-handed sequences. You can alter this limit using option `-H`.

Using option `-r`, you can specify a file that contains keyboard layouts that will be evaluated and displayed, without running an optimisation. In addition, using option `-V`, you can vary these layouts. For each layout in the file, all layouts are generated that differ from the given one by no more than the number of keys given by `-V`. If the keyboard layout in the file specifies a character using an uppercase letter, the position of this letter will not be varied. It is recommended to use `-m` to reduce output to the better ones of the generated variations, as otherwise the amount of generated layouts quickly becomes unmanageable.

5.2 Graphical output

Using option `-g` you can specify the name of a PostScript file to which graphics for all layouts are written that are also output as text. For example, using

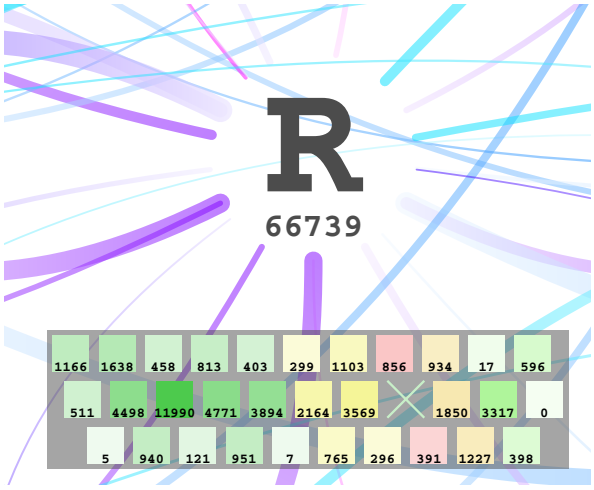
```
./opt -2 deutsch.txt -r bsptast.txt -g bsptast.ps
```

creates `bsptast.ps` with graphics for all the layouts stored in `bsptast.txt`, evaluated for German texts. Figure 1 shows an example.[1]

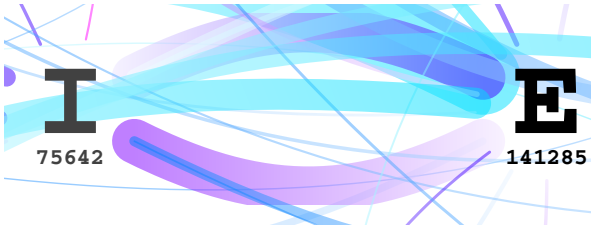
The graphics are made up of two layers. In the foreground, the letters are shown at key positions. Their blackness corresponds to their frequency. The frequency is printed as number below the letter (10000 corresponds to 10 %). Below the the letters, a miniature keyboard shows the distribution of subsequent keystrokes. The colour differs according to digram types, the colour intensity and the numbers on the miniature keys indicate the frequency of the digrams. Figure 2a gives an example. As you see, “r” and “R” account for about 6,7 % of key stokes. Following an “r” or “R”, the key struck most often is the one in the rest position of the left middle finger. In our example, “e” is mapped to this key, and about 1,2 % of all digrams are r-e.

The background visualises digrams by arcs. The thickness of an arc corresponds to digram frequency, the colour is different according to the type of digram. The order in which the keys in

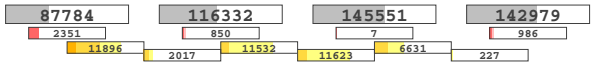
[1] To create the figure, the PostScript file was converted to PDF. This increases file size, however, in contrast to PostScript, PDF supports transparency.



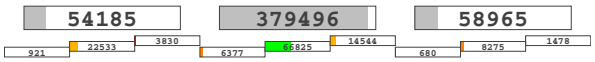
(a) Foreground: Individual keys and subsequent key strokes



(b) Background: Arcs indicating digrams



(c) Lower border: Finger load



(d) Sides (rotated): Groups and row jumps

Figure 2: Details of the graphical representation.

the digram are struck is indicated by arc curvature and the colour intensity shading. For the right hand, motions are counter clockwise, for the left hand, clockwise. In the interest of clarity, rare digrams and hand alternations are suppressed. Figure 2b displays thick, violet arcs joining “I” and “E”. In our example, “I” and “E” are entered by the left hand, so the upper arc belongs to i-e, and the lower arc to e-i.

The grey bars below the keyboard show the keystroke frequencies per finger. The frame around the bar corresponds to 25 % of all keystrokes, the number enclosed it is the relative frequency with respect to characters entered. For example, the grey box on the right of figure 2c shows that the left forefinger types approximately 14,3 % of the characters. This value is somewhat larger than in text output, as there, the number is referred to the number of keystrokes, and here, to the number of characters.

At the very bottom, you see little boxes with summed digram frequencies. The digrams are shown below the box for the finger that types the first key of the digram. Same finger repetitions are shown in red, where the three shades of colour intensity detail the distance the finger must jump in the same finger repetition. Adjacent finger strokes, separated into inward and outward motions, are shown in yellow-orange, with the intensity indicating row jumps of 0, 1, or 2. The size of the little boxes corresponds to 2 % of the total number of digrams, the enclosed number the relative frequency with respect to the total number of characters. For example, in figure 2c on the bottom left, you see that of all fingers of the left hand, the pinkie suffers the most same finger repetitions, namely about 0,24 per 100 characters.

Each keyboard row for each hand forms a *group*. The grey bars to the left and right of the keyboard show the cumulative keystroke frequency per group. The boxes around the bars correspond to 40 % of all keystrokes, the enclosed numbers are relative to the total number of characters. As you see in the middle grey box in figure 2d, about 37,9 % of all characters are entered on the middle row of the left hand.

The little boxes at the sides show digram frequencies without hand alternations, split according to the group into which the first keystroke goes. For each group, the graph further distinguishes according to the group into which the second keystroke goes. The size of the box corresponds to 20 % of the total number of digrams, the number in the box is the relative frequency with respect

to the total number of characters. For example, the number in the green box in the bottom centre of figure 2d means that for 100 characters, about 6,7 digrams have both keystrokes in the middle row on the left half of the keyboard.

When you create a graphics during an optimisation, be advised to use option `-i` to limit the number of iterations. If you abort the optimisation (using Ctrl+C), the graphics might be written incompletely.

The PostScript file created by the optimiser is human-readable and can be edited. At the beginning, you will find a number of switches to tune the visualisation. At the end, you will find the list of keyboard layouts to display.

5.3 Illustrate the typing process in the text

Using option `-M`, you can specify the name of an UTF-8 encoded text file, for which for each of the layouts in the layout file specified with option `-r`, it will be illustrated how the text is entered: Which hand is used, whether a character is located at the rest position, if it is an adjacent finger stroke or a same finger repetition with regards to the previous character. The result is written to an HTML file, the name of which is formed by appending `.html` to the name of the text file. At the beginning of this file, definitions are located that allow to customise the style of the presentation.

6 Changing the configuration

By default, the optimiser gets its settings from `standard.cfg`, a *configuration file* that comes with the optimiser. Using option `-K`, you can specify another configuration file. This option can be given multiple times. The effect is as if the configuration files specified this way would have been concatenated.

Configuration files are UTF-8 encoded text files and have a line-oriented structure. Empty lines are ignored. At the beginning of each line, an arbitrary number of spaces and tabulators can occur, which are ignored. If the first character after that is a #, the rest of the line will be ignored. Otherwise, a keyword is expected, followed by a keyword-dependent number of arguments which are separated by spaces or tabulators. After the arguments, a comment introduced with # is accepted. Arguments can be *names* (sequences of characters that that contain neither spaces nor tabulators), *strings* (sequences of characters in quotation marks, where the quotation mark introducing and ending the string must be equal, but are otherwise arbitrary), *flags* (+ or −), or numbers. Numbers can contain decimal comma or decimal point.

In the following, some mathematical symbols appear in parentheses, which will be referred to from Section 7.

6.1 Defining the character set

Usually, for an optimisation, only the most important characters in the languages for which the layout will be made need to be taken into account. Characters most often are given in pairs. The first character is on the base level, and the second one on the shifted level of the same key. If you specify more than two characters, the additional ones are treated as aliases of the second one. It is also possible so specify characters on their own, which then will get a key of their own. For example,

```
Zeichen 'aA'
Zeichen ' '
```

means that the lowercase and uppercase “A” share a key and the space character gets a key on its own. It is also possible to assign characters to a particular key. For example, using

```
FixesZeichen AD11 'ß'
```

will put “ß” on the key with the name AD11 (for more on the definition of keys and their names, see Section 6.2). The position of the “ß” will then not be subject to optimisation. However, its presence in the layout on AD11 can have a impact on which keys other characters will be mapped to.

The number of Zeichen and FixesZeichen in the configuration file must match the number of keys minus two (shift keys), see -DTASTENZAHL in Section 1. The same character must not occur in multiple Zeichen or FixesZeichen specifications.

Using Zeichen and FixesZeichen, optionally, one can specify a PostScript glyph name, which denotes the symbol which shall be used to represent the character in graphics. For example,

```
FixesZeichen SPCE ' ' underscore
```

specifies that space characters in the graphics will be represented by underscores. The specification of glyph names might be necessary for symbols outside of ISO-8859-1. To display them, in addition, the font used must support these symbols. The font can be selected by specifying PostScript font names using Zeichenfont and Beschreibungsfont, separately for the symbols contained in the layout and for the description (the numbers), for example:

```
Zeichenfont       FiraMono-Medium
Beschreibungsfont FiraSans-Book
```

Sometimes, you may wish to leave the base layer for a character in the shifted layer unoccupied. You can accomplish this by defining a placeholder character and mapping it to the base layer. For example, using

```
Platzhalter '¦'
Zeichen '¦ß'
```

the «ß» is placed in the shifted layer of a key the base layer of which is unoccupied. You can only define one placeholder, and must do so before all character definitions. For evaluation, the placeholder will be treated as if it would not appear in the corpus.

It is also possible to replace single characters by a sequence of other characters. For example, if you prefer to type “ss” instead of “ß”, you can specify this in the configuration file with

```
Ersatz 'ßss'
```

The first character in an Ersatz string is the character to be replaced, the rest the replacement. The replacement will only be used if the character to be replaced is not one of those given with Zeichen or FixesZeichen. The replacement can contain characters for which a replacement has been defined by a previous Ersatz string specification. Ersatz is of particular interest when you want to take into account dead key sequences in the optimisation.[2]

6.2 Defining the physical keyboard layout

The number of keys defined in the configuration file must correspond to what has been specified upon compilation, see -DTASTENZAHL in Section 1. An example for a key definition is

```
Taste AD03   4 1   3.25 1   -3   -   4   -
```

[2] totetasten.cfg contains an example.

Here, `AD03` is the name of the key.[3] Each name must be used only for a single key.

After the key name, the column and the row (ξ and ρ) of the key are given, in the example, `4` and `1`. These are both integers, rows in the range from 0 to 4 (where 0 is on top), and columns in the range from 0 to 15 (0 is on the left). These are followed by the horizontal and vertical coordinates (x and y) of the key, in the example `3.25` and `1`. These are arbitrary numbers, but will be typically be close to the values for column and row, respectively.

Next comes a number that indicates the finger (f) that strikes the key. Negative numbers are left, positive numbers are right. `−5` and `5` are the pinkies, `−4` and `4` the ring fingers, `−3` and `3` the middle fingers, `−2` and `2` the forefingers, and `−1` and `1` the thumbs. `0` is also a thumb, but it is left open, which one.

After that comes a flag. If it is `+`, the key is the rest position of the corresponding finger. The number following gives the individual key effort (α) of the key.

Finally, an optional flag can be specified, which is `+` or `−`, if for level selection, the left or right shift key is used with this key. If the flag is missing, for keys on a left finger the right shift key is assumed, and the other way round.

To define the left and the right shift key, use `ShiftL` and `ShiftR`. They expect the same arguments as `Taste`. The special keywords are used to make sure that both shift keys are actually specified in the configuration file.

6.3 Parametrising the evaluation

Individual keys

The individual key efforts are defined together with the keys, see Section 6.2. The target frequency is specified with `Zielhäufigkeit`. `Zielhäufigkeit` has ten arguments ($\tilde{\zeta}_{-5} \dots \tilde{\zeta}_{-1}, \tilde{\zeta}_1 \dots \tilde{\zeta}_5$), the unnormalised target frequencies of fingers, starting with the left pinkie, going over left and right thumb and ending with the right pinkie. Using `Fingerbelastung`, the corresponding ten weight factors ($\tilde{\phi}_{-5} \dots \tilde{\phi}_{-1}, \tilde{\phi}_1 \dots \tilde{\phi}_5$) are given. If the target frequency is exceeded, they will be used to compute an effort from the excess.

Digrams

There is a huge number of parameters for digrams. The most general one is `Bigramm`, which expects two key names and the effort ($\beta^{\mathrm{ex}}_{tt'}$), as a number. This effort is added to that computed from the other parameters.[4] For example:

```
Bigramm AD03 AB03 -2.5
```

Optionally, you can specify a string at the end of a `Bigramm` line. Digrams with the same string belong to the same digram type, and their cumulative frequencies appears in the text output. For example:

```
Bigramm AC04 AC07 0 'Symmetrical'
```

The same digram can appear in multiple `Bigramm` lines and, therefore, it can belong to multiple self-defined digram types.

[3] The key names in the packaged configuration files are taken from xkeyboard-config and are based on ISO 9995-1. If you like, you can replace them by more memorable names, for example, `AD01` by `Q`.

[4] Specification of individual digrams becomes awkward quickly. The included `gencfg.awk` demonstrates how to help yourself with a simple script.

The other parameters express efforts in terms of the terminology introduced in Section 3 and 5. The different classes of digrams are not disjoint. For example, each outward motion is a hand repetition as well. The efforts accounted for different classes are added.

`Handwechsel`, `Handwiederholung`, and `Auswärts` expect one number, which indicates the effort for a hand alternation (ω_1), a hand repetition (ω_0), and an outward motion ($\omega_{\leftrightarrow}$), respectively. A digram involving a thumb is never counted as an outward motion.

Using `DoppeltRabatt`, an effort for double strokes is computed by multiplying the argument (ϱ_0) with the difference of the individual key effort of the key in the rest position of the finger involved, and the individual key effort of the key itself. Usually, the difference is negative and, therefore, `DoppeltRabatt` usually results in an effort reduction. `ZeilenwiederholungRabatt` follows a similar idea. This keyword expects five numbers ($\varrho_1 \ldots \varrho_5$). When both key stokes are made with different fingers (excluding thumbs) of the same hand, both key stokes are on the same row but not on the middle row (2), the first key of the digram is not both on the lower line and typed with the pinkie, then, for the second key of the digram, a difference as described above is computed, multiplied with one of the arguments of `ZeilenwiederholungRabatt`, and the result is taken as an effort. The argument is selected according to the difference in the columns of the keys: Difference 1 means first argument, difference 2 second argument, and so on. For example,

```
ZeilenwiederholungRabatt 0.5 0.25 0.16666667 0.125 0.1
```

defines that the effort reduction for row repetition is inverse proportional to the difference of the columns.

Using `KollisionKonstant` and `KollisionDistanz` allows to define efforts for same finger repetitions. Both keywords expect five arguments ($\kappa_1 \ldots \kappa_5$ and $\varkappa_1 \ldots \varkappa_5$, respectively), one for each finger, starting with the thumb and going outwards. The effort of a same finger repetition is the value from `KollisionKonstant` plus the product of the value from `KollisionDistanz` and the distance the finger must jump in the same finger repetition. The distance is computed from the coordinates (x, y) of the keys, see Section 6.2. For example,

```
KollisionKonstant  10 10 10 10 10
KollisionDistanz   10 10 10 10 10
```

defines, among others, that the efforts for same finger repetitions are independent of the finger.

Using `Nachbar`, the efforts for adjacent finger strokes are defined. There are four arguments ($\nu_{3/2}, \nu_{5/2}, \nu_{7/2}, \nu_{9/2}$), for the pairings of thumb/forefinger, forefinger/middle finger, middle/ring finger, and ring finger/pinkie, respectively. For example, with

```
Nachbar  0 1.3333333 2 4
```

adjacent finger stokes are considered to require the more effort, the farther outside the fingers involved are.

Finally, for hand repetitions without involvement of the thumbs, efforts of hand distorting digrams are accounted for, specified with `SchrägZS` and `SchrägYX`. The keywords both expect two numbers (θ_{-1}, θ_1 and ϑ_{-1}, ϑ_1, respectively), which are applied to the left and right hand, respectively. To obtain an effort, the value from `SchrägZS` is multiplied with the ratio of row difference and column difference, the one from `SchrägYX` with the ratio of vertical and horizontal distance of the two keys. For example,

```
SchrägZS         0   1
SchrägYX         1   0
SchrägNenner0 0.1 0.1
```

causes left hand distorting digrams to be evaluated according to coordinates, and right hand distorting digrams to be evaluated according to column and row of the keys. Furthermore, we have used `SchrägNenner0` to specify two numbers (Δ_{-1} and Δ_1, for the left and the right hand, respectively) which will be added to both the column difference and the horizontal distance in this computation.

Shift digrams

The effort of a shift digram is obtained from the effort of the key digram it consists of by multiplication with one of the two arguments (ς_+ and ς_-) of `Shiftbigramm`. The first one is used if the effort of the key digram is positive, the second one if it is negative.

Trigrams

The most general definition of trigram efforts is possible using `Trigramm`. It works like for `Bigramm`, however, with three instead of two key names.

For trigrams the first and third key of which are operated with the same hand, additional efforts are accounted for. First, the effort for the key digram made up by the first and third key is determined. It gets multiplied with one of the two values (τ_+ and τ_-) specified with `Indirekt`: The first one, if the effort is positive, and the second one, if it is negative. If the middle key is typed by another hand, the product is taken as an additional effort, and, furthermore, the value (ω_{11}) given with `Doppelwechsel` is added. If all three keys are typed with the same hand, the product is compared with the sum of the digram efforts for the first/second and second/third key. If the product is larger, the additional effort is the product minus the sum plus the value (ω_{00}) given with `Doppelwiederholung`. Otherwise, the effort is the value specified with `Doppelwiederholung`.

Finally, for trigrams made up of an outward and an inward motion, the value ($\omega_{\rightleftarrows}$) specified with `Wippe` is added to the effort.

Similarity and confusability

The most general definition of the confusability is with `Verwechslungspotenzial`. It works like `Bigramm`. Additionally, you can specify contributions to the confusability for keys typed with the same (`VPKollision`, η) or adjacent fingers (`VPNachbar`, υ). Using `VPSymmetrisch`, you can specify contributions (Σ) for keys that are typed with the same finger of different hands, and by using `VPSymmetrischGleicheZeile`, you can specify contributions ($\Sigma_=$) for keys that are additionally in the same row. `VPHandwechsel` (ϖ) applies to keys that are operated by different hands.

Using `Ähnlich`, the similarity (ε) of two or more characters is defined. For example,

```
Ähnlich 'bp' 0.1
Ähnlich 'sz' 0.05
```

means that “b” and “p” are twice as similar as “s” and “z”.

Preferences

Using `Vorliebe`, preferences for the mapping of characters to keys can be defined. The keyword expects a string of characters for which the same set of keys is preferred, a number (λ) that is the larger, the stronger the preference is, and the set of keys, given as a list of key names. For

example, to express that you would like to see x, c, and v on the four keys on the lower left, you could use:

```
Vorliebe 'xcv' 0.1  AB01 AB02 AB03 AB04
```

Missing symbols

Using `Fehlt`, the effort (Φ) for entering symbols that are not part of the layout can be specified. For example:

```
Fehlt 100
```

means, that for each symbol that appears in the corpus but is not part of the layout, an effort of 100 is accounted for.

7 Computing efforts, in formulas

We group the symbols in the layout $\mathbb{B}$ in n pairs, one symbol for the base level and one for the shifted level. Each symbol $z \in \mathbb{B}$ can be represented by a pair of indices, $z = (p_z, e_z)$, where $p_z \in \{0 \dots n-1\}$ and $e_z \in \{0, 1\}$. For the n symbol pairs, we need n keys and, additionally, two shift keys for level selection. The symbol keys are numbered starting from zero, $t \in \{0 \dots n-1\}$. The shift keys get the numbers n and $n+1$.

We consider a keyboard input i as a combination of a symbol key t_i and level selection e_i, represented by the index pair $i = (t_i, e_i)$. In what follows, i, j and k will denote such index pairs, and summation over these indices are performed over all symbol keys and levels. By t_i, we denote the first component (that is, the key) of the index pair i, by e_i, we denote the second component (the level). Finally, by s_i, we denote the shift key that belongs to key t_i.

A layout π is a permutation of $(0 \dots n-1)$. It maps a keyboard input i to a symbol $z = \pi_i = (\pi_{t_i}, e_i)$. Thereby, symbols are associated with keys always in pairs. The level a particular symbol is located on is kept fixed.

In the following, δ is the Kronecker delta, $\bar{\delta}_{ij} = 1 - \delta_{ij}$ its complement, Θ the step function

$$\Theta(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases},$$

$\bar{\Theta}(x) = \Theta(-x)$ and $(x)_\pm = \pm x \Theta(\pm x)$ are the positive and the negative part of x, respectively.

7.1 Frequencies

The texts to be typed are given by a number of corpora which we number by an index $c \in K$. Corpus c has a weight g_c which is given by normalisation of the weights specified on the command line (using `-G`), such that $\sum_c g_c = 1$. Each corpus is given in the form of tables with symbol frequencies $H_i^{(c)}$, digram frequencies $H_{ij}^{(c)}$ and trigram frequencies $H_{ijk}^{(c)}$.

Let $\mathbb{F}$ be the set of all non-fixed fingers, let f_t be the finger operating t. The number of symbols in c typed with non-fixed fingers is $N^{(c)} = \sum_{f' \in \mathbb{F}} \sum_i \delta_{f' f_{t_i}} H_i^{(c)}$. The number of key strokes made with non-fixed fingers for c is $T^{(c)} = \sum_{f' \in \mathbb{F}} \sum_i (\delta_{f' f_{t_i}} + e_i \delta_{f' f_{s_i}}) H_i^{(c)}$. The relative, weighted total

frequencies are

$$h_i = \sum_{c\in K} g_c H_i^{(c)}/N^{(c)}\,, \tag{1a}$$
$$h_{ij} = \sum_{c\in K} g_c H_{ij}^{(c)}/N^{(c)} \quad \text{and} \tag{1b}$$
$$h_{ijk} = \sum_{c\in K} g_c H_{ijk}^{(c)}/N^{(c)}\,. \tag{1c}$$

The relative frequency of all symbols that are contained in the corpus, but not in the layout, is

$$\bar{h} = \frac{\sum_{c\in K}\sum_{z\notin\mathbb{B}} g_c H_z^{(c)}}{\sum_{c\in K}\sum_{i\in\mathbb{B}} g_c H_i^{(c)}}\,.$$

7.2 Effort

The total effort is the sum of several contributions,

$$A(\pi) = A_0(\pi) + A_1(\pi) + A_2(\pi) + A_3(\pi) + A_\mathrm{f}(\pi) + A_\mathrm{fehlt}\,. \tag{2}$$

In the text output, for the quantity “total effort”, $100A$ is printed.

The contibution to the total effort from typing individual keys is

$$A_1(\pi) = \sum_i a_i h_{\pi_i}\,, \tag{3}$$

where $a_i = \alpha_{t_i} + e_i(\alpha_{s_i} + \beta_{s_i t_i})$ and the α are the per-key efforts as specifed in the configuration file. In the text output, for the quantity “positional effort”, $100A_1$ is printed.

The contibutions to the total effort from typing symbol digrams and symbol trigrams are

$$A_2(\pi) = \sum_{ij} a_{ij} h_{\pi_i\pi_j}\,, \tag{4}$$
$$A_3(\pi) = \sum_{ijk} a_{ijk} h_{\pi_i\pi_j\pi_k}\,, \tag{5}$$

where

$$a_{ij} = (1-e_j)\Big[\beta_{t_i t_j} + e_i\varsigma(\beta_{s_i t_j})\Big] + e_j\Big[\beta_{t_i s_j} + \gamma_{t_i s_j t_j} + e_i\varsigma(\beta_{s_i s_j})\Big]\,, \tag{6}$$
$$a_{ijk} = (1-e_j)\Big[(1-e_k)\gamma_{t_i t_j t_k} + e_k\gamma_{t_i t_j s_k}\Big]\,, \tag{7}$$

see also Table 1 and 2. The β and γ are the efforts for key digrams and key trigrams,

$$\begin{aligned}
\beta_{tt'} = {}& \beta^\mathrm{ex}_{tt'} + \delta_{1|f_t-f_{t'}|}\nu_{|f_t+f_{t'}|/2} + \delta_{tt'}\varrho_0(\alpha_{\Gamma_t} - \alpha_t) + \\
&+ \Theta(f_t f_{t'})\,\bar{\delta}_{tt'}\Big[\omega_0 + \Theta(|f_{t'}| - |f_t|)\,\bar{\delta}_{1|f_t|}\omega_{\leftrightarrow}\Big] + \bar{\Theta}(f_t f_{t'})\,\omega_1 + \\
&+ \Theta(f_t f_{t'})\,\bar{\delta}_{f_t f_{t'}}\bar{\delta}_{1|f_t|}\bar{\delta}_{1|f_{t'}|}\delta_{\rho_t\rho_{t'}}\bar{\delta}_{\rho_t 2}\left(1 - \delta_{3\rho_t}\delta_{5|f_t|}\right)\varrho_{|\xi_t-\xi_{t'}|}(\alpha_{\Gamma_{t'}} - \alpha_{t'}) + \\
&+ \lim_{\delta\to\Delta_{\chi t}} \Theta(f_t f_{t'})\,\bar{\delta}_{f_t f_{t'}}\bar{\delta}_{1|f_t|}\bar{\delta}_{1|f_{t'}|}\left(\frac{\theta_{\chi_t}|\rho_t - \rho_{t'}|}{|\xi_t - \xi_{t'}| + \delta} + \frac{\vartheta_{\chi_t}|\gamma_t - \gamma_{t'}|}{|x_t - x_{t'}| + \delta}\right) + \\
&+ \bar{\delta}_{tt'}\delta_{f_t f_{t'}}\left(\kappa_{|f_t|} + \varkappa_{|f_t|}\sqrt{(x_t - x_{t'})^2 + (y_t - y_{t'})^2}\right)
\end{aligned}$$

and

$$\gamma_{tt't''} = \gamma^{\mathrm{ex}}_{tt't''} + \Theta\left(f_t f_{t''}\right)\left\{\bar{\Theta}\left(f_t f_{t'}\right)\left(\omega_{11} + \beta^{\mathrm{ind}}_{tt''}\right) + \right.$$
$$\left. + \Theta\left(f_t f_{t'}\right)\left[\omega_{00} + \Theta\left(\beta^{\mathrm{ind}}_{tt''}\right)\left(\beta^{\mathrm{ind}}_{tt''} - \beta_{tt'} - \beta_{t't''}\right)_+ + \Theta\Big(\left(f_t - f_{t'}\right)\left(f_{t''} - f_{t'}\right)\Big)\omega_{\rightleftarrows}\right]\right\},$$

where $\chi_t = \Theta\left(f_t\right) - \bar{\Theta}\left(f_t\right)$, $\beta^{\mathrm{ind}}_{tt'} = \tau_+\left(\beta_{tt'}\right)_+ - \tau_-\left(\beta_{tt'}\right)_-$ and $\varsigma\left(x\right) = \varsigma_+\left(x\right)_+ - \varsigma_-\left(x\right)_-$. Γ_t is the key in the rest position of f_t. Keys with the same and row and column are considered identical, $\delta_{tt'} = \delta_{\rho_t\rho_{t'}}\delta_{\xi_t\xi_{t'}}$.

When $\zeta_f = \tilde{\zeta}_f / \sum_{f'}\tilde{\zeta}_{f'}$ is the normalised target frequency for f as obtained from the $\tilde{\zeta}_f$ in the configuration file, contibution to the total effort from finger load is

$$A_{\mathrm{f}}\left(\pi\right) = \sum_{c\in K} g_c \sum_{f\in\mathbb{F}} \phi_f \left(b^{(c)}_f\left(\pi\right) - \zeta_f\right)^2_+ , \tag{8}$$

where $b^{(c)}_{f'}\left(\pi\right) = \sum_i (\delta_{f'f_{t_i}} + e_i\delta_{f'f_{s_i}})H^{(c)}_{\pi_i}/T^{(c)}$ is the load on finger f' for corpus c, and $\phi_f = \tilde{\phi}_f$ if the evaluation is done without and $\phi_f = \left(1+\tau_+\right)\tilde{\phi}_f$ it is done with considering trigrams.

The non-mechanical criteria are made up from confusability and preferences,

$$A_0\left(\pi\right) = \sum_{t=0}^{n-1}\sum_{t'=0}^{n-1} \varphi_{tt'}\varepsilon_{\pi_t\pi_{t'}} - \sum_{t=0}^{n-1}\lambda_{\pi_t t}\,. \tag{9}$$

φ denotes the confusability of two keys,

$$\varphi_{tt'} = \varphi^{\mathrm{ex}}_{tt'} + \bar{\delta}_{tt'}\delta_{f_t f_{t'}}\eta + \delta_{1|f_t - f_{t'}|}\upsilon + \left[1 - \Theta\left(f_t f_{t'}\right)\right]\varpi +$$
$$+ \bar{\Theta}\left(f_t f_{t'}\right)\bar{\delta}_{1|f_t|}\bar{\delta}_{1|f_{t'}|}\delta_{|f_t||f_{t'}|}\left(\bar{\delta}_{\rho_t\rho_{t'}}\Sigma + \delta_{\rho_t\rho_{t'}}\Sigma_=\right),$$

ε the similarity of two symbols, and λ the preference for mapping a particular pair of symbols to a particular key.

The effort for symbols missing from the layout is $A_{\mathrm{fehlt}} = \bar{h}\Phi$.

7.3 Pareto property

The property that the layout that minimises A is a Pareto opimum with respect to to the individual corpora $c \in K$ is due to the fact that the efforts $A^{(c)}$ sum up to the total effort,

$$A = \sum_{c\in K} g_c A^{(c)}\,. \tag{10}$$

If the optimum π_0 would not be a Pareto optimum, there would by a layout π_1 and a $c_0 \in K$ with $A^{(c_0)}\left(\pi_1\right) < A^{(c_0)}\left(\pi_0\right)$ and $A^{(c)}\left(\pi_1\right) \le A^{(c)}\left(\pi_0\right)$ for $c \neq c_0$. But this would imply that the sum would become smaller, which contadicts the assumtion that π_0 is an optimum.

The sum property (10) for the finger load is explicit in Equation (8). A_1, A_2 and A_3 (Equation (3), (4) and (5)) are linear in the h and, therefore, it is possible to pull out the c summation from the definition (1) of the h, for example $A_1\left(\pi\right) = \sum_i a_i \sum_c g_c H^{(c)}_{\pi_i}/N^{(c)} = \sum_c g_c \sum_i a_i H^{(c)}_{\pi_i}/N^{(c)} = \sum_c g_c A^{(c)}_1\left(\pi\right)$. Finally, A_0 in Equation (9) is independent of the corpus and, therefore, $A_0\left(\pi\right) = A^{(c)}_0\left(\pi\right)$ and the sum property holds due to $\sum_c g_c = 1$.

If, instead, one single corpus is used which is assembled externally form different partial corpora, the nonlinearity of Equation (8) in h prevents us from giving the guarantee that the total optimum is Pareto optimal with respect to those partial corpora.

7.4 Statistical error

Initially, we consider one single corpus. We imagine that the corpus is a sample from an infinite, ideal corpus, and that we created it by repeatedly selecting a random word from this ideal corpus, where each choice is independent from the others. This is the crucial simplification. In reality, the uncorrelated units are presumably something larger than individual words.

Let the probability to pick the word w for at particular selection be p_w. The frequency with which the word w is selected is binomially distributed, with an average value of $\mu_w = p_w H_{\text{wort}}$ and a variance of $\sigma_w^2 = H_{\text{wort}} p_w (1 - p_w)$. We can estimate these parameters from the observed frequencies as $\mu_w = H_w$ and $\sigma_w^2 = H_w (1 - H_w/H_{\text{wort}})$, respectively, where H_w is the frequency of word w and H_{wort} is the size of the sample.

We denote the frequency with which the symbol i and symbol digram ij occur in w by w_i and w_{ij}, respectively. The number of symbols in a word is $|w| = \sum_i w_i$, and the number of keystrokes that are required to enter it is $\|w\| = \sum_i (1 + e_i)\, w_i$. With this conventions, we can write the number of symbols and keystrokes in our sample as $N = \sum_w |w| H_w$ and $T = \sum_w \|w\| H_w$, respectively, the relative frequency of symbol i as $h_i = \sum_w w_i H_w / N$, and so on.

The frequencies of words in a sample will deviate from their expectation values. The deviation $\delta A(\pi^{-1})$ from the expectation value of the effort $A(\pi^{-1})$ of the keyboard layout π^{-1} caused by these deviations δH_w is

$$\delta A(\pi^{-1}) = \sum_w \left[\sum_i a_{\pi_i} \frac{w_i - |w|\, h_i}{N} + \sum_{ij} a_{\pi_i \pi_j} \frac{w_{ij} - |w|\, h_{ij}}{N} + \right.$$
$$\left. + \sum_{f' \in \mathbb{F}} 2 \phi_{f'} \left(b_{f'} - \zeta_{f'}\right)_+ \sum_i (\delta_{f' f_{t_{\pi_i}}} + e_i \delta_{f' f_{s_{\pi_i}}}) \frac{w_i - \|w\| \bar{h}_i}{T} \right] \delta H_w \,,$$

where $\bar{h}_i = H_i / T$ and the finger load and the total number of symbols have been linearised around the expectation values of the word frequencies. As we consider the frequencies of different words as independent, $\langle \delta H_w \delta H_{w'} \rangle = \sigma_w^2 \delta_{ww'}$ holds, where $\langle \cdot \rangle$ denotes an expectation value. With this relation,

$$\left\langle \delta A^2(\pi^{-1}) \right\rangle = \sum_w \left[\sum_i a_{\pi_i} \frac{w_i - |w|\, h_i}{N} + \sum_{ij} a_{\pi_i \pi_j} \frac{w_{ij} - |w|\, h_{ij}}{N} + \right.$$
$$\left. + \sum_{f' \in \mathbb{F}} 2 \phi_{f'} \left(b_{f'} - \zeta_{f'}\right)_+ \sum_i (\delta_{f' f_{t_{\pi_i}}} + e_i \delta_{f' f_{s_{\pi_i}}}) \frac{w_i - \|w\| \bar{h}_i}{T} \right]^2 H_w \left(1 - \frac{H_w}{H_{\text{wort}}}\right) .$$

For evaluation, the contents of the square brackets are multiplied out. From the products of the terms $(w_i - |w|\, h_i)/N$, $(w_{ij} - |w|\, h_{ij})/N$ and $\left(w_i - \|w\| \bar{h}_i\right)/T$, six correlation terms arise, for example

$$\sigma_{ij;k}^{(2\text{f})} = \sum_w \frac{w_{ij} - |w|\, h_{ij}}{N} \frac{w_k - \|w\| \bar{h}_k}{T} H_w \left(1 - \frac{H_w}{H_{\text{wort}}}\right) .$$

They can be directly computed from the corpus, and be used to compute $\langle \delta A^2 \rangle$ straightforwardly,

$$\left\langle \delta A^2(\pi^{-1}) \right\rangle = \ldots + 2 \sum_{ijk} a_{\pi_i \pi_j} \sum_{f' \in \mathbb{F}} \phi_{f'} \left(b_{f'} - \zeta_{f'}\right)_+ (\delta_{f' f_{t_{\pi_k}}} + e_i \delta_{f' f_{s_{\pi_k}}}) \sigma_{ij;k}^{(2\text{f})} + \ldots$$

To compute the variance of the difference of efforts, the π-dependent prefactors are replaced by differences for the two layouts π and π', for example, $a_{\pi_i \pi_j}$ is replaced by $a_{\pi_i \pi_j} - a_{\pi'_i \pi'_j}$. The total variance for multiple corpora is obtained from the individual variances, weighted with g_c^2.

8 Appendix

8.1 The included frequency files

The German corpus, on which the files `deutsch.txt.*` are based on, is composed as follows:

- 636 kB of randomly chosen lines from the German Leipziger Korpus, post-processed by Karl Köckemann. The original corpus contains 3 million sentences, chosen from different newspapers. Karl's modifications adapt it to the new German orthography.
- 642 kB from essays published in the magazine c't between 22/1999 and 12/2006. These essays use the new German orthography and were available as HTML. The HTML-formatting has been removed, as well as head lines and paragraphs containing `<a>`, `<code>`, or similar.
- 150 kB non-fiction and 200 kB fiction texts from Project Gutenberg:
  Arnold Sommerfeld, Relativitätstheorie, from Deutsches Leben in der Gegenwart; Sigmund Freud, Massenpsychologie und Ich-Analyse, Chapter I und II; Mihai Nadin, Jenseits der Schriftkultur, 40 kB from Volume I, Chapter I; Alexander Lipschütz, Warum wir sterben (ca. 40 kB); Gerhart Hauptmann, Bahnwärter Thiel (ca. 20 kB from the beginning); Stefan Zweig, Brennendes Geheimnis (ca. 20 kB from the beginning); Robert Walser, Der Gehülfe (ca. 20 kB from the beginning); Gottfried Keller, Die Leute von Seldwyla, Vol I (ca. 20 kB from the beginning); Franz Kafka, Die Verwandlung (ca. 20 kB from the beginning); Alfred Döblin, Die Ermordung einer Butterblume und andere Erzählungen (ca. 20 kB from the beginning); Hermann Hesse, Siddhartha (ca. 20 kB from the beginning); Thomas Mann, Der Tod in Venedig (ca. 20 kB from the beginning); Rainer Maria Rilke, Zwei Prager Geschichten (ca. 20 kB from the beginning).
  These texts use the old German orthography. Only "daß" has been replaced by "dass", and paragraphs have been merged into lines.

For the files `deutsch-t.txt`, the hyphenation patterns `hyph-de-1996.pat.txt` (version of May 2014) and same corpus have been used.

The English corpus on which the files `englisch.txt.*` are based on is composed as follows:

- 920 kB of randomly chosen lines from the English Leipziger Korpus. American spelling partly has been changed to English (for example, "center" in "centre"). Some units have been changed to something internationally meaningful (for example, "miles" to to "kilometres")
- 417 kB non-fiction text:
  - From O'Reilley open books: Stephen L. Talbott, The future does not compute, Chapter 23, filtered as the c't essays.
  - From Project Gutenberg:
    Charles Darwin, On the Origin of Species, Chapter I; James Clerk Maxwell, Five of Maxwell's Papers.
  - From the Open American National Corpus, directory `written_2/non-fiction/OUP`:
    Abernathy, Chapter 3; Berk, Chapter 7; Fletcher, Chapter 10; Kauffmann, Chapter 5; Rybczynski, Chapter 1.
- 339 kB fiction text from Project Gutenberg:
  Lewis Carroll, Alice in Wonderland; Joseph Conrad, Heart of Darkness, Chapter I; W. Somerset Maugham, Of Human Bondage, Chapter 1–16; James Joyce, Ulysses, about 2500 lines.

For the files `englisch-t.txt`, the hyphenation patterns `hyph-en-gb.pat.txt` (version of May 2010) and the same corpus have been used.

8.2 Example layouts

For more information on the layouts in `bsptast.txt`, see: Arensito, Aus der Neo-Welt, Colemak, Klausler, KOY, Neo 2. Partially, I have added the umlauts myself.

8.3 Further information

The optimiser has a home page, which provides its latest version and additional information. A good starting point to learn more about the background on which judgements of layouts are based is `http://adnw.de` (provided that you know German).