# QWERTY vs. Dvorak Efficiency: A Computational Approach

Ricard Torres*
ITAM and Universitat de Girona

This version: June 2013

**Abstract**

After David's pioneering contribution, many authors have claimed that the prevalence of the QWERTY keyboard layout is an instance of lock-in on an inferior technology, because the Dvorak Simplified Keyboard is more efficient. But other authors, notably Liebowitz and Margolis, have disputed this claim. Here we proceed to examine the relative efficiency of the two layouts by computational means. The result is that Dvorak is actually more efficient than QWERTY. Our computations depend on a coefficient that measures the additional effort associated with a displacement of the hand. For intermediate values of this parameter, the gain in efficiency exceeds 10%.

## 1 Introduction

In an influential article, Paul David [4] argued that sometimes the path of technological progress is not driven solely by gains in efficiency, but that particular characteristics of the way in which new technologies were introduced could substantially affect future developments, even resulting in the prevalence of inferior technologies. In order to illustrate the point, David mentioned the QWERTY keyboard layout[19] that is still dominant for the

*Centro de Investigación Económica, Instituto Tecnológico Autónomo de México (ITAM), and Universitat de Girona. Address: Faculty of Economics and Business, Campus Montilivi, Universitat de Girona, E-17071 Girona, Spain. Email address: ricard.torres@udg.edu.

English language [18] (and derivations of it dominate for many other languages too).

Since its introduction, it has been argued by various authors and is still being argued[1] that an alternative keyboard layout patented in 1932, the so-called Dvorak Simplified Keyboard [16] is more efficient. David cites some experiments performed by the US Navy in the 1940s by which typists were able to match their performance with QWERTY after ten days of retraining with the new layout. He goes on to explain that the Dvorak layout has never been able to gain widespread usage due to what economists have named later "network effects": since the QWERTY layout is used by most firms, people who learn have an interest in being trained in this layout; analogously, since most people are trained in the prevailing layout, firms have no interest in investing in the alternative one.

W. Brian Arthur [1] developed a model that highlights the role of increasing returns (that may be a consequence not only of network effects, but also of learning) for history dependence. His paper was among many that have used the QWERTY vs. Dvorak efficiency as an illustration of such phenomena.

Later, Liebowitz and Margolis [8] contested David's argument. They argued that all experiments that had been performed in order to compare the relative efficiency of the two keyboard layouts were badly designed. In particular, they unravelled the fact that the US Navy experiment had been performed under the supervision of August Dvorak himself, who had an economic interest at stake, and this made the entire experiment suspect.

Hossain and Morgan [5] report a series of experiments in which they test whether a QWERTY-like effect of being locked into an inferior technology can occur. In their experimental set-up this never happened, so they add to the skepticism raised by Liebowitz and Margolis.

Recently, Kay [6] raises again the issue about the efficiency of QWERTY. Kay settles some historical issues, and next uses probabilistic arguments in order to clarify allegations that have been made about the layout. In particular, he justifies that the design of QWERTY, in pursuing the objective of avoiding key jams, was mostly guided by the principle of setting as continguous letter pairs that are infrequent in the English language, rather than,

[1] Two instances from the "blogosphere" are: Dana Albert (http://www.albertnet.us/2009/06/defendants-i-type-lot.html), and Marcus Brooks (http://dvorak.mwbrooks.com/dissent.html).

as is often argued, trying to separate letter pairings that are frequent. Kay shows how Dvorak would have caused substantially more key jams than QWERTY, which was an issue until the introduction of the IBM golf-ball electrical typewriter in 1961.

In this paper, we raise again the basic question of the relative efficiency of the QWERTY vs. Dvorak layouts. The experiments that have been performed in the past have been complicated by the fact that there is a nonnegligible learning curve in order to realize people's efficiency in typing. One important element, that was already pointed out by David as favouring lock-in, is that maximum efficiency in typing seems to be associated with touch typing [20]. This allows us to take a computational, rather than experimental, approach in order to evaluate the question of efficiency. We calculate an approximation to the "effort" that it would take a touch-typist to type a given text using the two layouts. The results are unambiguous: Dvorak is always more efficient than QWERTY. Our results show that, for intermediate values of the effort parameter (see later), the gain in efficiency is of about ten percent.

## 2 Calculating efficiency

Our objective is to select a particular text file and compute (an approximation to) the typing effort it will take to type it with each of the two keyboard layouts.

The basis of our computations is the touch-typist left and right hand positions and the movements necessary in order to type each succeeding glyph. Taking as unit of effort the fact of typing a key when the finger is positioned above that key, we approximate the total typing effort by a linear function which adds to that the effort associated with hand displacement. For each successive glyph, the hit count is increased by one unit corresponding to hitting the current key, plus a given parameter (between 0 and 1) times the number of rows the hand must traverse. Of course, this has a degree of arbitrariness, and ideally it should be based on a physical study of the actual effort exerted in each case. But, in a first approach, we dispense with this requirement by resorting to a *robustness* argument: if the efficiency ranking is the same regardless of the effort parameter we associate with hand displacements, then we may be satisfied that our computational approach makes sense *qualitatively*. Of course, this could and should be complemented

by a quantitative approach based on actual physical measurements.

Our interest is in computing the effort of typing using a modern computer keyboard. The first problem we encounter is that, even within the US QWERTY standard, keyboard layouts differ[19]. For instance, the key that types '\' and '|' depending on the shift state, is sometimes positioned in the same row as the digits, usually in the next row downwards, and other times in the one below. So the first thing we do is to fix, for both QWERTY and Dvorak, a layout which we have found to be the most common one. The details are in our code, which is included in Appendix A.

We use the `python` (`http://www.python.org/`) language,[2] because it is simple and freely available for different computer platforms.

For touch typing purposes, the keyboard is divided into two sides, left and right, each of which is assigned to the corresponding hand. The keys used by both layouts are positioned in 4 rows from top to bottom; the top row contains the digits and some symbols, and the three bottom rows contain the letters and the remaining symbols. There is a fifth row that has a special treatment and, for our purposes, just contains the space bar; this key is hit with the thumb of either hand, and we assume that it does not require any hand displacement, no matter what the current hand position is. The carriage return key we assume that is accessed via either the 2nd or 3rd row. The 4th row contains at each extreme the shift keys. For simplicity, we ignore the possibility of using the "Caps Key" in order to type succeeding capital letters; this might easily be incorporated into our program, but we just thought it of little relevance. Another simplification we use is to ignore all nonascii characters, even gramatically correct ones like those written with a diaeresis,[3] or long hyphens: this is of little relevance for our purpose, as can be seen by the figures we report at the end of the program run, and would require resorting to "internationalized" keyboards that allow typing letters with diacritics, which is outside the scope of this work.

[2] We have used version `2.7.3`, which, at the time of this writing, is considered the status-quo, hence standard, version (`http://wiki.python.org/moin/Python2orPython3`). We try to adhere to the style guide (`http://www.python.org/doc/essays/styleguide.html`), except that, for convenience, we use 3 spaces for indentation instead of the recommended 4.

[3] The *New Yorker* magazine still uses the spellings "coördination" and "coöperation," instead of the more usual ones without any diacritics.

## 2.1 The texts used

It has been asserted (see Kay [6]) that Twain's *Life on the Mississippi* [14] (LOTM) was the first book typed using the new mechanical typewriter. Kay uses this text for some of his numerical exercises, so we will use it as well. We obtained the text file from Project Gutenberg (`http://www.gutenberg.org/`), and, for the purposes of our calculations, we edited out the Project's additions at the beginning and end of the file. A second text we also used (again, following Kay [6]) is Adam Smith's *Wealth of Nations* [13] (WON), which we took from the same source and to which we applied the same treatment. So we have two pre-20th century English texts, one in the American variety and the other in the British one.

For purposes of comparison, we also took two modern publicly available texts. The first is the Economic Report of the President 2012 [3] (ERP). We used the online text version of this report [3], skipping the statistical appendices; we edited the different files taking out the `html` headers and footers, and concatenated them.[4] For another modern text that is neither Literature nor Economics, we chose Eric S. Raymond's *The Cathedral and the Bazaar* [11] (TCATB). We took the online `html` files, converted them to text files using the "print" option of the freely available `lynx` program (`http://lynx.isc.org/`), and concatenated them.

## 2.2 Results

Recall that the "cost" added by each new glyph typed is the sum of two components: the fix (normalized) unitary component derived from hitting a key, plus an additional component that measures the effect of having to displace the hand from its previous position (if the shift key is depressed, then both hands intervene, so we do this for each of them). The addition that corresponds to hand displacement is the product of the number of rows the hand must traverse times a fixed coefficient which we set between 0 and 1.

The key parameter in the computation of the total hit count corresponding to each keyboard layout is the coefficient that multiplies the number of rows traversed. If this coefficient is set to 0, then the total hit count is the

[4] We took the 2012 Report because the conversion to text of the one corresponding to 2013 was buggy at the time we downloaded it (June 2013).

| Text | Lines | Characters | Ignored | QWERTY | Dvorak | % Gain |
|---|---|---|---|---|---|---|
| LOTM | 14808 | 823096 | 77 | 1096114.0 | 993183.0 | 10.36 |
| WON | 35203 | 2257006 | 797 | 2993136.5 | 2683391.5 | 11.54 |
| ERP | 9846 | 587317 | 12 | 825221.0 | 738707.0 | 11.71 |
| TCATB | 2178 | 117230 | 218 | 161417.0 | 146613.5 | 10.09 |

Table 1: Layout efficiency when using a coefficient 0.5.

| Text | 0.2 | 0.4 | 0.6 | 0.8 |
|---|---|---|---|---|
| LOTM | 4.66 | 8.60 | 12.00 | 14.95 |
| WON | 5.17 | 9.57 | 13.38 | 16.69 |
| ERP | 5.29 | 9.74 | 13.54 | 16.81 |
| TCATB | 4.53 | 8.38 | 11.69 | 14.58 |

Table 2: Efficiency gain of Dvorak over QWERTY for different coefficient values.

same for both layouts, and just equals the total number of characters (including the carriage return). Let us first present the results for the different texts when we set this coefficient at 0.5. Later on we will report the effect of changing this value.

The results we present in Table 1 contain, for each text used, the following data: the number of lines entered, the total number of characters read, the number of those that are ignored (because they are nonascii), the total hit count for the two keyboard layouts, and the percent gain in efficiency that the better layout (which happens to be always Dvorak) represents over the other. We may see that, when the coefficient of cost per row traversed is set to 0.5, the Dvorak layout has an efficiency gain above 10% with respect to the QWERTY layout.

In Table 2, we report the results of the sensitivity analysis when the coefficient of cost per row traversed varies between 0.2 and 0.8, with step 0.2. We can see that the efficiency gain is monotonic with respect to the coefficient.

# 3 Discussion

To the extent that the method used here to compute the relative efficiency of the two keyboard layouts seems reasonable, the results point unambiguously toward superiority of the Dvorak layout. Traditionally, people have emphasized the importance of contiguity of pairs, because key jamming in mechanical typewriters happened to be highly dependent on this. The method we use, based on touch typing, cares about the proximity of the keys that are more often used, and not so much about pairs per se. From this viewpoint, maybe the "Ideal" keyboard layout alluded to by David [4] might turn out to be even more efficient than Dvorak. But that is besides the point we want to make here: we are not interested in finding an optimal keyboard layout, but in finding whether the often made claim that the Dvorak layout is more efficient than the QWERTY one can be substantiated. And it turns out that, if our method is well founded, it is, contrary to the doubts about this point that were raised by Liebowitz and Margolis [8].

Now, one possible objection to our method is that most people who use keyboards do not touch-type. While this is certainly true, we just want to point out that the polemic about the relative efficiency of the two layouts was framed from the beginning in terms of how fast *expert typists* could type with them.

## 3.1 Is the QWERTY lock-in a rare occurrence?

Liebowitz and Margolis [8] and [9]) base their argument on the fact that, if there is efficiency to be realized, market forces should ultimately push toward the adoption of the most efficient technology. In particular, they question the importance of network effects. However, the relevance in the short run of network effects is illustrated by the numerous examples and strategic recommendations displayed by Shapiro and Varian [12] in their book about business strategy in the "new economy."

For very short time frames, history dependence is almost a triviality. For example, it seems clear that the fact that a large percentage of public funding for research is devoted to defense, shapes the type of technologies that are being developed: that is most likely the reason why, currently, flying drones (unmanned aerial vehicles) have been operational for a time while electric cars still represent a fringe market. Another element that surely influences the path and type of technological development is patent law.

The problem is not whether there is path-dependence, but whether *in the long run* the path of technological development shapes the type of technologies that end up prevailing. Liebowitz and Margolis [9] rightly point out that most authors cite over and over again the same examples of QWERTY and the VHS vs. Betamax standards battle. If the phenomenon were as prevalent as some authors imply, shouldn't there be plenty more examples to show?

One issue here is that history dependence is a much wider topic than lock-in on inferior technologies. For example, a consequence of history dependence might be that the pace of advancement of the electronics and information industries has been much faster than that of the energy generation industry.[5] Phenomena of this type seem quite difficult to measure empirically, but we may expect empiricists to develop ingenous means of testing once they focus on the problem.

But, concentrating on the issue of technological lock-in, we should observe that there is something very peculiar about the QWERTY layout case. If we view it as a technological means to achieving an end, namely to type text, there has been little technological change since the advent of electronic typing. And this has contributed to the fact that the issue has not being long superseded by technological innovation, as are the standards for tape recording. In fact, the confluence of several developments will probably cause obsolescence of the QWERTY issue in the near future: first, the widespread use of mobile devices with virtual (small) keyboards; second, the increasing sophistication and use of voice as an input device.

## 4 Conclusions

In this paper, we tackle directly the problem of comparing the relative efficiency of the QWERTY and the Dvorak keyboard layouts, from the viewpoint of a touch-typist. The results, which depend on a coefficient that measures the extra effort of hand displacement, favor unambiguously the Dvorak layout. For intermediate values of the coefficient, Dvorak is approximately 10% more efficient than QWERTY. However, as Kay [6] points out, this gain in efficiency should apply only after the introduction of electronic means of typing in the 1960s. But, with this proviso, this means that

[5] It is claimed that, nowadays, the backwardness of this sector is retarding development in fields as different as computing and aviation.

the prevalence after the 1960s of the QWERTY layout can be seen as an illustration of lock-in on an inferior technology.

Ideally, the effort coefficient should be the result of physical measurement. However, there is no reason to think that our linear specification corresponds to reality or, even if that were true, that the coefficient is the same for all initial hand positions and keys. Our work here is intended merely as an approximation, and its justification stems not from how exact an approximation it is, but from its robustness to changes in the parameters.

## References


[1] Arthur, W. Brian, "Competing Technologies, Increasing Returns, and Lock-In by Historical Events," *The Economic Journal* (1989), 99: 116–131.

[2] Arthur, W. Brian, "Comment on Neil Kays paper—'Rerun the tape of history and QWERTY always wins' " *Research Policy* (2013), in press, http://dx.doi.org/10.1016/j.respol.2013.01.012.

[3] Council of Economic Advisers, *Economic Report of the President 2012*, http://www.gpo.gov/fdsys/browse/collection.action?collectionCode=ERP.

[4] David, Paul, "Clio and the Economics of QWERTY," *American Economic Review Proceedings* (1985), 75: 332–337.

[5] Hossain, Tanjim, and John Morgan, "The Quest for QWERTY," *The American Economic Review* (2009), 99: 435–440.

[6] Kay, Neil, "Rerun the tape of history and QWERTY always wins," *Research Policy* (2013), in press, http://dx.doi.org/10.1016/j.respol.2013.03.007.

[7] Kay, Neil, "Rerun the tape of history and QWERTY always wins: Response to Arthur, Margolis, and Vergne," *Research Policy* (2013), in press, http://dx.doi.org/10.1016/j.respol.2013.03.010.

[8] Liebowitz, Stan, and Stephen Margolis, "The Fable of the Keys," *Journal of Law and Economics* (1990), 22: 1–26.

[9] Liebowitz, Stan, and Stephen Margolis, "Network Externality: An Uncommon Tragedy," *Journal of Economic Literature* (1994), 8: 133–150.

[10] Margolis, Stephen, "A tip of the hat to Kay and QWERTY," *Research Policy* (2013), in press, http://dx.doi.org/10.1016/j.respol.2013.03.008.

[11] Raymond, Eric, *The Cathedral and the Bazaar*, O'Reilly Media, Sebastopol (CA), 2001. There is a publicly available version at: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

[12] Shapiro, Carl, and Hal Varian, *Information Rules: A Strategic Guide to the Network Economy*, Harvard University Press, 1998.

[13] Smith, Adam, *An Inquiry into the Nature and Causes of the Wealth of Nations*, Project Gutenberg, http://www.gutenberg.org/cache/epub/3300/pg3300.txt, last accessed June 2013.

[14] Twain, Mark, *Life on the Mississippi*, Project Gutenberg, http://www.gutenberg.org/cache/epub/245/pg245.txt, last accessed June 2013.

[15] Vergne, Jean-Philippe, "QWERTY is dead; long live path dependence," *Research Policy* (2013), in press, http://dx.doi.org/10.1016/j.respol.2013.03.009.

[16] Wikipedia, "Dvorak Simplified Keyboard," http://en.wikipedia.org/wiki/Dvorak_Simplified_Keyboard, last accessed June 2013.

[17] Wikipedia, "Eric S. Raymond," http://en.wikipedia.org/wiki/Eric_Raymond, last accessed June 2013.

[18] Wikipedia, "Keyboard layout," http://en.wikipedia.org/wiki/Keyboard_layouts, last accessed June 2013.

[19] Wikipedia, "QWERTY," http://en.wikipedia.org/wiki/QWERTY, last accessed June 2013.

[20] Wikipedia, "Touch typing," http://en.wikipedia.org/wiki/Touch_typing, last accessed June 2013.

# Appendix

## A The `python` code

Regarding the program, it should be taken into account that in `python` *indentation is essential*, and in particular it determines the extent of the code that is executed within `if` conditionals, and `for` and `while` loops.

The file with the code, as well as the different text files used, are available from the author upon request.

```
#!/usr/bin/env python
# qwerty.py: given an input ascii text file, determine which keyboard
#    layout would be more efficient typing it, QWERTY or Dvorak, and
#    print some statitics about it.

# Author: Ricard Torres <ricard.torres@udg.edu>, June 2013

## First, two settable parameters:

# The first, the additional effort for each row that must be traversed
# Should be a number between 0 and 1
addition_per_row = 0.5;

# The second, the name of the file from which we will compute the
# hit count for each keyboard:
input_text_file = "twain-life-on-the-mississippi.txt";

# Now, define the keyboards (qwerty, dvorak) we are going to use
# Each keyboard definition is a vector with as many components as rows
# There are 5 rows from top to bottom (indexed from 0 to 4)
# For each row, the keyboard specifies two strings:
#   The first (indexed by 0) corresponds to no shift key (lowercase),
#   The second (indexed by 1) corresponds to shift key active (uppercase).
# The 5th row, for our purposes, contains just the space bar.
#
# Note: US keyboards contain one more key in row 1 (the second), whereas
# in European keyboards this additional key is usually found in row 2
```

```
qwerty_keyboard_us = [ [ ’‘1234567890-=’,’~!@#$%^&*()_+’ ],  # 13 keys
 [ ’qwertyuiop[]\\’ , ’QWERTYUIOP{}|’ ],                     # 13 keys
 [ ’asdfghjkl;\’’, ’ASDFGHJKL:\"’ ],                         # 11 keys
 [ ’zxcvbnm,./’ , ’ZXCVBNM<>?’ ],                            # 10 keys
 [ ’ ’, ’ ’] ] ;                                             # 1 key (sp bar)

dvorak_keyboard_us = [ [ ’‘1234567890[]’,’~!@#$%^&*(){}’ ],
 [ ’\’,.pyfgcrl/=\\’ , ’\"<>PYFGCRL?+|’ ],
 [ ’aoeuidhtns-’, ’AOEUIDHTNS_’ ],
 [ ’;qjkxbmwvz’ , ’:QJKXBMWVZ’ ],
 [ ’ ’, ’ ’] ] ;

# Valid characters for us are unicode characters with values
# in the ascii range, from 0x0020 (32, space) till 0x007e (126, tilde),
# a total of 95 values (the same we have in both keyboards)
#
# So we define a vector with 95 values indexed from 0 to 94,
# in which each value contains the character with decimal
# value ’index + 32’:

valid_chars = [];
for i in range(32,127):
   valid_chars.append(chr(i));

# In order to reference a character from its decimal ascii number, just
# need to specify: valid_chars[number-32]
# which, of course, is equivalent to chr(number)
# Conversely, from an index in valid_chars the decimal ascii value
# is found by adding 32

# Initialize various counting variables:
qwerty_hit_count = 0;
dvorak_hit_count = 0;
total_chars = 0;
ignored_chars = 0;
total_lines = 0;
```

```
# Begin main loop, whose objective is to compute the hit count
#    for each of the two keyboards

# This variable is an indicator of the current keyboard
current_is_qwerty = 1;
# We use it in order to avoid counting twice common parameters,
#   like the total number of lines or chars

for keyboard in [qwerty_keyboard_us, dvorak_keyboard_us ]:

# First, determine the position of characters in the keyboard
# For each (valid) character, 'character_position' is a vector with
# three components: side, row, and shift state
# The side can take the values: 0 (left) and 1 (right)
# The row can take the values 0 to 3 (top to bottom)
# The shift key state can take the values 0 (not set) and 1 (set)
#
# Actually, the space key (32) has a special treatment, and we
# set its side to center (2), row to 4, and shift to 0
# (because it's irrelevant)
#
# The definition of 'character_position' can be viewed as a sort of
#  inverse of the keyboard definition

   character_position = [];
# Space character has a special treatment
   character_position.append([2,4,0]);
# Initialize the rest
   for validch in range(1,len(valid_chars)):
      character_position.append([]);

   for row in range(4):
      for shift_state in [0,1]:
         for keybch in range(len(keyboard[row][shift_state])):
            for validch in range(1,len(valid_chars)):
               if keyboard[row][shift_state][keybch] == valid_chars[validch]:
                  character_position[validch] = [0,row,shift_state];
```

```
                    if (row == 0 and keybch > 5) or (row > 0 and keybch > 4):
                       character_position[validch][0] = 1; # right hand

# The current position of the hands has two components [ left, right ], and
# each can take the value of any row between 0 and 3
# We assume: (1) shift key is equivalent to row 3
#            (2) space key does not modify previous row (thumbs are used)
#            (3) carriage return corresponds to right hand -- rows 1 or 2
#            (4) Caps key not used: succeeding capital letters entered individually
# Initialize:
   current_hand_position = [ 2, 2 ];

# Initialize hit count variable:
   hit_count = 0;

   f = open(input_text_file,'r',1);
   myline = f.readline();
   while myline:
      if current_is_qwerty:
         total_lines += 1;  # increase line count on qwerty
      for mychar in myline:
         if current_is_qwerty:
            total_chars += 1;  # increase char count on qwerty
         numchar = ord(mychar) ; # easier to work with decimal representation
         if current_is_qwerty and ( numchar & 0x80 ): #8th bit set: nonascii
            ignored_chars += 1;
            # Optionally write message for nonascii utf8 or other enc chars
            # print "Line num:",total_lines,
            # print "Will ignore char",numchar ;
         elif numchar == '10': # linefeed: hit carriage return
            jump = min( abs(current_hand_position[1] - 1),
                        abs(current_hand_position[1] - 2));
            new_row_is_2 = 1;
            if ( jump < abs(current_hand_position[1] - 2) ):
               new_row_is_2 = 0;
            hit_count += 1 + jump * addition_per_row;
            current_hand_position[1] = 1;
            if new_row_is_2: current_hand_position[1] = 2;
```

```
         elif numchar == 32: # space: hit space bar
            hit_count += 1;
         elif numchar >= 33 and numchar <= 126: # valid ascii: hit char
            [side,row,shift] = character_position[numchar-32];
            jump = abs( current_hand_position[side] - row );
            hit_count += 1 + jump * addition_per_row;
            current_hand_position[side] = row;
            if shift: # Use other hand for shift key
               jump = abs( current_hand_position[1-side] - 3 );
               hit_count += 1 + jump * addition_per_row;
               current_hand_position[1-side] = 3;
         else:  # Ignore all other characters (CR=13, control, nonascii)
            pass;
      myline = f.readline(); # Read new line and continue while loop
# End of while loop:
# we have finished reading the file, so first we close it
   f.close();
# Assign the hit count to the corresponding keyboard
   if current_is_qwerty:
      qwerty_hit_count = hit_count;
   else:
      dvorak_hit_count = hit_count;
# Change the keyboard index
   current_is_qwerty -= 1;
# This is the end of the for loop, so now we just print the overall result
# Print final result
print "\nInput file:",input_text_file;
print "Total number of lines read:",total_lines;
print "Total number of characters read:",total_chars;
print "Of which",ignored_chars,"were ignored";
print "\nHit count is:"
print "QWERTY:", qwerty_hit_count;
print "DVORAK:", dvorak_hit_count;
print "Addition per row traversed:", addition_per_row;
# Determine the more efficient keyboard
qwerty_is_more_efficient = 0;
if qwerty_hit_count <= dvorak_hit_count:
   qwerty_is_more_efficient = 1;
```

```
print "The more efficient keyboard is",
if qwerty_is_more_efficient:
   print "QWERTY";
else:
   print "DVORAK";
difference = abs(qwerty_hit_count - dvorak_hit_count);
ratio = difference/min(qwerty_hit_count , dvorak_hit_count);
print "Hit count difference:",difference;
print "Percent gain:",ratio*100;
# This is the end of the program
```