BLICK: a phonotactic probability calculator (manual)

Bruce Hayes
Department of Linguistics
UCLA

July, 2012

Quick start guide

1. Find a Windows machine
2. Download the program and unzip it.
3. Click on the program, BLICK.exe.
4. Left side window is for individual words, right side is for a file name with words in it.
5. Valid transcription format is displayed on the interface.
6. If you need more help read the rest of this manual.

Contents

1. Function and purpose of this program

Input: any string of English phonemes, expressed with a symbol set provided. It can be a real word or one you made up.

Output: a numerical value, inversely reflecting the phonotactic "goodness" (probability, well-formedness, word-likeness…) of the word. For more on the meaning of this score, see below.

Examples: the current version of BLICK predicts that *ket* [kɛt] should a completely perfect word of English (penalty score zero), that *doit* [dɔɪt] should be a somewhat peculiar word of English (score 3.094), and that *nguhyee* [ˈŋʌji] should be a pretty horrible word of English (score 12.295).

Purpose: providing a reference point for phonotactic probability in experimentation. This is the same purpose as the pioneering phonotactic model proposed by Vitevitch and Luce (2004) and discussed in section **Error! Reference source not found.** below.

2. Installation

Windows only … sorry. I hope to have a more widely usable version soon.

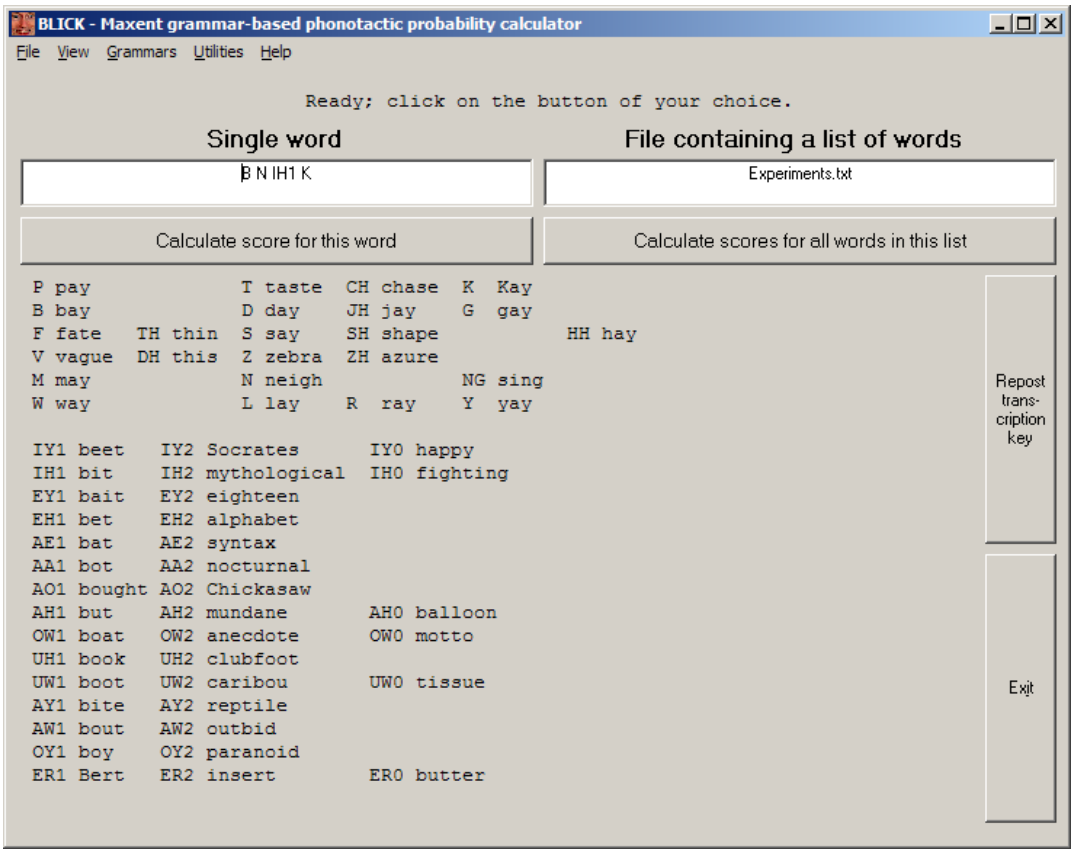
I believe that BLICK does not actually need to be "installed". Instead, just download the zip file BLICK.zip, put it where you want it on your computer, and unzip it. This will produce a small folder structure, of which the topmost folder contains the program. It is called **BLICK.exe**. Click on it to run.

This *should* work; if it does not, please report to me (Bruce Hayes, bhayes@humnet.ucla.edu) and we can try a more sophisticated installation procedure.

[continued on next page …]

3. Running BLICK

When you click on the BLICK program, you should get a user interface that looks like this:



This interface provide a number of options.

- In the text box on the left, you can **type a word**, using the phonetic transcription system shown with examples on the interface. Then, and by clicking on the button labeled **Calculate score for this word**, you can ask BLICK to calculate its score. Higher numbers are worse.

- You can make a **list of words**, likewise using the phonetic transcription system shown, and ask BLICK to calculate the scores of all of them. Put this list in the **Input** folder, daughter of the main folder. To process the list, click on the button that says **Calculate scores for all words on this list**.

    - Here is the **format for the input file**: It should be in plain text. Each word should be on its own line and employ the transcription system shown on the interface. The word may optionally be followed by comments; put them on the same line as the word, coming after it and separated from it by a tab. Separate different bits of your comment with further tabs. Example:

```
B IH1 N TH   <TAB>  Experiment Bailey and Hahn 2001 0.416440292
B L EH1 M P  <TAB>  Experiment Bailey and Hahn 2001 0.525604929
B L EH1 S K  <TAB>  Experiment Bailey and Hahn 2001 0.487800176
B R EH1 L CH <TAB>  Experiment Bailey and Hahn 2001 0.361621749
B R EH1 L TH <TAB>  Experiment Bailey and Hahn 2001 0.322596198
B R EH1 N TH <TAB>  Experiment Bailey and Hahn 2001 0.437273626
```

Where to find your results: BLICK will put its results in an output file located in the **Output** folder. If you input file is named **MyFile.txt**, then your output file will be named **BLICKOutputForMyFile.txt.** Open this file with a spreadsheet program.

4. How BLICK calculates phonotactic probability

4.1 Prologue

BLICK is somewhat complicated. It is entirely possible to use BLICK without understanding anything about how it works, and since experimentalists have so many other things they have to worry about, I would refrain from snickering if you do this!

The bottom line is really how well BLICK models existing experimental data on phonotactic well-formedness — i.e., how well it does when benchmarked. For this, you can skip ahead to section 5 below.

4.2 Basis of BLICK

BLICK is based on phonology, the field of linguistics that studies sound patterns in language. Much of the phonology included in BLICK represents ideas and analytical insights that are decades old, but the program also includes some more recent theoretical developments that help make its behavior approximate that of real English speakers.

In brief, the BLICK system makes use of a **maxent grammar** consisting of a set of **constraints**, each bearing a **weight**. BLICK itself is simple, because it is merely a grammar-applicator. When BLICK evaluates a word, it assesses how the word violates the constraints in the grammar, then computes the summed product of violations and constraint weights to form the penalty score that is output by the program.

4.3 How can I interpret the scores output by BLICK?

There is a mathematically well-defined interpretation of these scores. If you negate them, and take *e* to the result, you get a **likelihood value**. This is a number that is proportional to probability; more specifically, the share that the word receives out of a total probability value of one, which is allocated amongst the (vast) set of possible strings of speech sounds (taken from the symbol inventory) that has the same statistical distribution of lengths as the corpus of data (included) on which the BLICK grammar was trained.

The probability interpretation matters, because it forms the basis of how BLICK computes constraint weights. However, for most practical applications it suffices just to use the raw scores output by BLICK, interpreting them as penalty scores.

4.4 What is a constraint?

Intuitively a constraint is a phonological formula; when a phonologist proposes a constraint, she claims that words that violate the constraint are phonotactically worse. The constraint formula consists of a sequence of **feature matrices**, each which expresses a class of speech sounds sharing the properties denoted by the features. Here is one of the constraints in the default grammar used by BLICK:

$$*[+\text{labial}]\begin{bmatrix}-\text{consonant}\\+\text{labial}\end{bmatrix}$$

This means, literally, "It's bad when a labial sound precedes a nonconsonantal labial sound." In reader friendlier terms, the [+labial] sounds are (in IPA) [p, b, m, f, v, w]. So, when BLICK encounters a word like *Puerto Rico* (as pronounced in Spanish-like fashion as [pwɛɹtoɹiko], it "sees" the [pw] sequence and assesses a violation.[1] Here is the line-up of constraint and wordform:

p	w	ɛ	ɹ	t	o	ɹ	i	k	o
|	\								

$$*[+\text{labial}]\begin{bmatrix}-\text{consonant}\\+\text{labial}\end{bmatrix}$$

It is also possible for there to be multiple violations, whenever there are multiple locations within a word that match up with the constraint.

4.5 Constraint weights

The grammar of constraints in BLICK is weighted by a statistical procedure that maximizes its fit to the data of English, here represented by a corpus (included) of 18,033 existing English word forms.[2] The fitting algorithm is known (by mathematical proof) to converge to the best possible fit, defined as that which assigns the greatest possible probability to the observed data; i.e. that of the corpus. By doing this, the algorithm deflects probability away from words that are highly un-wordlike in English. Specifically, if the constraint system is set up correctly, such words will violate constraints that have been assigned very high weights.

---

[1] Notice that many speakers of English Anglicize the word as [ˌpɔrto ˈriko], escaping the violation.

[2] Compounds, forms created by productive affixation, and unassimilated loanwords have been excluded from the corpus. For discussion of the corpus and how it was formed, see Hayes and White (in press). The version in BLICK has been subjected to a modest amount of further editing, removing transcription errors and flagging further affixed forms.

The constraint-weighting algorithm can also accomplish more subtle effects. For instance, experimental work has shown that English words having statistically unusual sounds like [θ], [ɔɪ], [ʒ] are a bit "off" phonotactically and are treated as less word-like than words with frequent sounds like [k], [ɛ], or [t] — this should be intuitively clear if you say to yourself the non-words [kɛt] and [θɔɪʒ]. If we include in the grammar constraints that simply penalize words containing the rare sounds, the resulting constraints will receive relatively moderate weights that appropriately assign positive, but not enormous, penalty scores to words containing them.

4.6 The constraint system

While it has been possible to set the weights of the BLICK grammar on a purely algorithmic basis, this is not so for the constraint system; the current state of the art yields grammars that are both too verbose for our purposes, and also somewhat problematic in their predictions; see in particular Hayes and White (in press).

The constraints I selected from BLICK can be examined in the spreadsheet that comes with the program, which provides a certain amount of discussion as well. I found these constraints partly by reading the research literature in phonology, and partly by trial and error. There are basically three kinds.

4.6.1 Constraints governing the sonority system

These are the constraints that require that the edges of syllables to bear appropriate sonority profiles — [pla] is better than [lpa], [alp] is better than [apl]. This is an old notion in phonology and has recently been the subject of extend psycholinguistic experimentation. For BLICK, I found it possible to implement a very simple strategy: simply include all logically possible sonority constraints in the grammar (see Hayes (2012) for how this was done) and let the weighting algorithm decide which ones are important. Some of them actually were weighted at zero, essentially deleting them from the grammar.

A nice benefit of this strategy is the BLICK grammar accomplishes "sonority projection": just like real speakers of English, it prefers moderate sonority violations like [bda] to flagrant ones like [lda] (Berent et al 20xx, Berent et al. 20xx), even though both types have zero frequency in the training data. For discussion of this point, see Daland et al. (2011), Hayes (2012).

4.6.2 Constraints against individual segments

Recall from above that there is reason to penalize rare segments like [θ] and [ɔɪ]. For this purpose the grammar used by BLICK includes constraints against every single sound in the inventory. Indeed, it actually uses twice as many such constraints, for it employs for each sound a constraint against the sound occurring in the onset of a syllable (i.e. the consonant or consonant cluster that precedes the vowel) and a constraint against the sound occurring in the coda (i.e. any consonant(s) following the vowel). Some of these constraints actually get very high weights, for the sound [ŋ] cannot occur in onsets at all, nor can [h] occur in codas. In other cases, lower

weights produce the gentle skewing of probability values in the direction of words composed exclusively of frequent sounds. In still others, the maxent weight system assigned a zero weight (because the segment is actually better represented than we would expect, given the rest of the grammar) and for convenience I removed the constraint in question.

4.6.3 Remaining constraints

The remainder of the constraints were created by hand. The procedure I followed was as follows essentially, "keep fixing the grammar until it no longer allows bad things". More specifically:

- I made lists of pseudoforms containing all possible sequences of a particular kind:
  - all possible two-consonant onsets
  - all possible two-consonant codas
  - all possible two-consonant intervocalic sequences
  - all possible two-vowel sequences

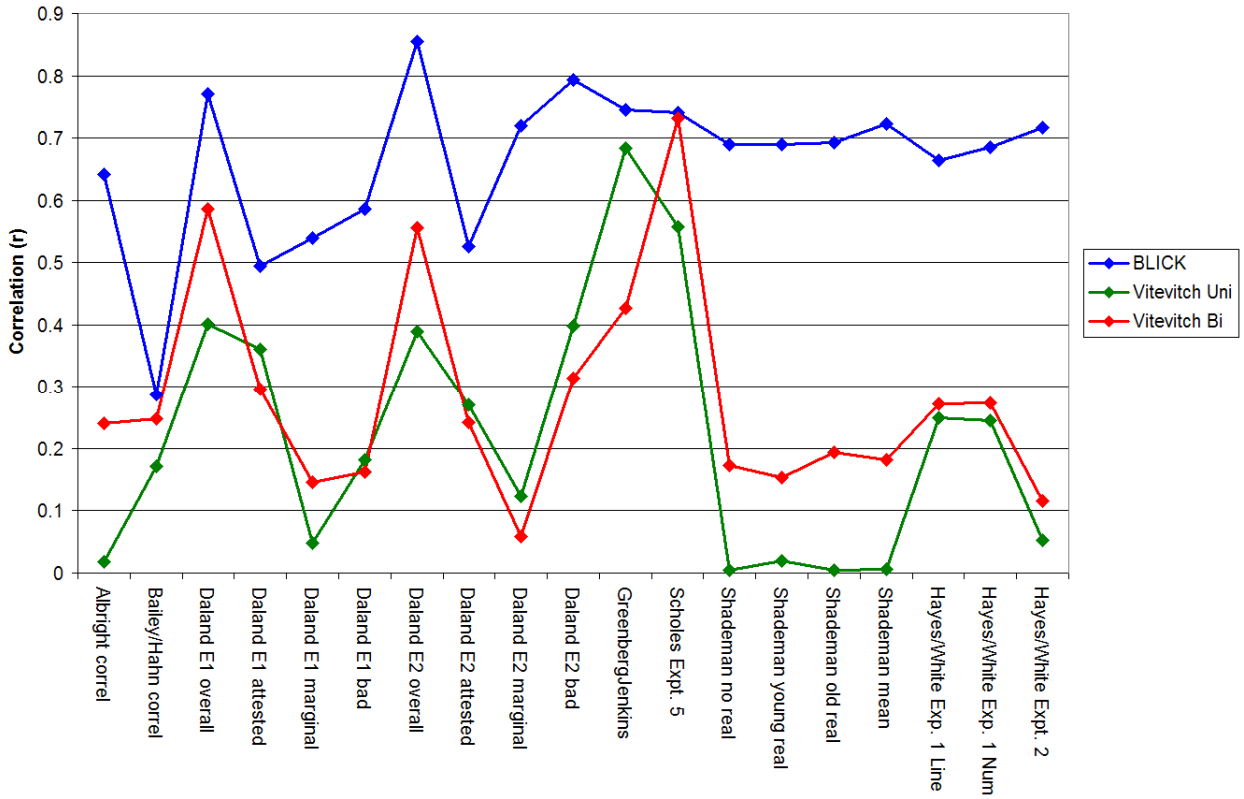
  In each case I identified the sequences that actually occur in the corpus.
- I also made a long list of "phoneme salad" words; i.e. segment sequences picked at random.
- I repeated ran these lists through my draft grammars, and augmented them until they had stopped assigning good scores to bad-sounding sequences.
- When I created constraints, I tried to make them as general as possible (applying to as many sequences as possible).

The grammar used by BLICK is about as large as can conveniently be run on the software I used to create it (software by Colin Wilson, and downloadable from http://www.linguistics.ucla.edu/people/hayes/Phonotactics/). It would be possible to add new segmental-level constraints, but more constraints on stress location would have to be evaluated in overnight computer runs, at least with 2012-era desktop computers.

5. Benchmarking BLICK

I've tried to benchmark BLICK against experimental data, comparing it with alternatives. At least in the case of the various ratings studies I've checked, I've gotten reasonably good correlations between the ratings assigned by BLICK and those obtained experimentally from speakers of English. These correlations appear to be higher than those obtained for the widely-used and pioneering Phonotactic Probability Calculator. This testing is described an article in progress that I hope to submit (and web-post) shortly.

At present, I can offer a comparison with the current state-of-the-art public phonotactic probability calculator, that proposed by Vitevich and Luce (2004) and usable online at http://www.people.ku.edu/~mvitevit/PhonoProbHome.html. I checked the correlation of the two program's predictions against seven ratings experiments. BLICK always gets a higher correlation, sometimes substantially so. The following graph summarizes the situation:



For the various studies, consult references below.

It can be seen that sometimes the Vitevitch model does almost as well as BLICK. This happens when the consonant-vowel pattern of the experiment is fixed, as it was (in the form CCVC) in both Greenberg and Jenkins (1964) and Scholes (1966). In fact, my impression is that the studies done by Prof. Vitevich and his colleagues has likewise sensibly restricted the consonant-vowel pattern, typically to CVC. For discussion of why the model would do well here, and poorly elsewhere, see Hayes (2012, ms.).

In some cases, BLICK does not do as well as the model that was included in the paper from which I got the data; for instance, Albright's (2009) phonotactic model outperforms BLICK on Albright's (2009) data. However, without having tested all models against all data, I feel that BLICK can be credited with versatility; it performs fairly well on a wide range of data. For instance, unlike the model in Hayes and Wilson (2008), BLICK can distinguish subtle difference among words that are basically well-formed (see discussion in Daland et al, 2011). And unlike the model of Coleman and Pierrehumbert (1997), which outperforms it on some of the Shademan (2007) data, it can distinguish between truly horrible zero-frequency onsets (like in [lba]) and merely bad ones (like in [dba]). I hope to report further model comparison in future work.

6. What other background ought I to know to use BLICK in a maximally informed way?

These are listed going from beginner material to more advanced work.

- For English phonetic transcription, I recommend the following two textbooks: Ladefoged (20xx) and Rogers (2001), in the References below.

- For phonological features and how they are used form constraints, I recommend my phonology textbook, Hayes (2009). The features used in BLICK (included in the program folder) are similar to, but not identical with, the ones in this textbook.

- For maxent grammars and their role in phonotactics: Goldwater and Johnson (2003), Hayes and Wilson (2008).

7. Why is the program called BLICK?

The reference is to Chomsky and Halle (1965), who pointed out that *brick* [bɹɪk] is real word of English, *blick* [blɪk] is not a real word but could be (because it is phonotactically fine), and *bnick* [bnɪk] is not a real word and could not be. Occasionally researchers refer to a phonotactic ratings study as a "blick test".

8. Extra utilities and functions

If you click on the menus you will extra little functions I programmed into BLICK.

8.1 Tableau

The "tableau" (terminology from Optimality Theory, in phonology) is a table that lists all the words in rows, all the constraints in columns, and the number of constraint violations for each word/constraint combination in the main body of the table. To create a tableau, first go to the **Utilities** menu, then click on **Include a tableau** (a check will appear by it), then click the **Calculate scores for all words** button. Your tableau will appear in the Output folder under the name **TableauForFileName.txt**. It is a plain text file, tab-delimited.

8.2 List of forms violating constraints

This is also a table. The rows are constraints, and each constraint is followed by various annotations (including the number of forms in the input file that violate it), then by all of the forms that violate it. To create a violation list, first go to the **Utilities** menu, then click on **Include a list of words** (a check will appear by it), then click the **Calculate scores for all words** button. Your violation list will appear in the Output folder under the name **ListOfViolatingWordsForFileName.txt**. It is a plain text file, tab-delimited. For large input files, some constraint will have a very large number of columns; whether you can read all of them depends on the size and your spreadsheet program. If this is a problem for you try using the tableau utility (immediately above) instead.

8.3 Neighbor counts

BLICK can count (and list) lexical neighbors. It counts as a neighbor any word that differs by exactly one substitution, insertion, or deletion from the target word. Tou can either find the

neighbors for one single word, or for a list in an input file. Output file is named **NeighborhoodsForMyFile.txt**. To access this capacity, click on suitable items in the **Utilities** menu.

8.4 Vitevitch replication

This a partial replication of the well-known Phonotactic Probability Calculator, described by Vitevitch and Luce (2004) and available at Prof. Vitevitch's website (http://www.people.ku.edu/~mvitevit/PhonoProbHome.html). The model works by grouping unigrams or bigrams into "slots" going across the words from left to right ("first unigram, second unigram, …" or "first bigram, second bigram, …"), computing a word-frequency-weighted frequency value average for each slot-filler, and summing across the word.

Access the Vitivich replication in BLICK from the **Utilities** menu. It works just like the main routines of BLICK; you can either run it for one single word or for an input containing multiple words. In the latter case, your output file will be named **VitevitchPPCOutputForMyFile.txt**.

The BLICK version of the Vitevitch system yields numbers that are *similar*, but not identical, to the original. The reason is that BLICK is based on a different English phonetic lexicon and transcription system. The bigram part of the model is currently yielding different outcomes from the Vitevitch original; so use with caution.

9. Feedback

Please send feedback, including bug reports, to Bruce Hayes at bhayes@humnet.ucla.edu.

The source code, written in Microsoft Visual Basic 6, is posted on the program web site.

10. References *(not done)*

Berent, Iris,et al. (20xx) on sonority projection; *Cognition*
Chomsky, Noam and Morris Halle (1965) *Journal of Linguistics*
Daland, Robert et al. (2011) *Phonology*
Greenberg and Jenkins (1964) *Word*
Hayes, Bruce (2009) *Introductory Phonology.* Blackwell Wiley.
Hayes (2012) sonority projection. ICPhS, also Hayes web site
Hayes and White (in press) *Linguistic Inquiry*; also Hayes web site
Hayes and Wilson (2008) *Linguistic Inquiry*; also Hayes web site
Ladefoged, Peter *A Course in Phonetics,* 5th ed.
Rogers, Henry (2001) *The Sounds of Language*, Longman.
Scholes, Robert (1966) *Phonotactic Probability*. The Hague: Mouton.
Vitevitch, Michael S. and Luce, Paul A. (2004) A web-based interface to calculate phonotactic probability for words and nonwords in English. *Behavior Research Methods, Instruments, and Computers*, 36, 481-487.