MASSACHUSETTS INSTITUTE OF TECHNOLOGY
A. I. LABORATORY

Artificial Intelligence
Memo No. 218

August 1971

INFORMATION THEORY AND THE GAME OF JOTTO

Michael Beeler

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0002.

The word game, JOTTO, has attracted the interest of several computer programmers over the years, not to mention countless devoted players. Rules are:

1. Each of 2 players selects a 5-letter English word, not a proper noun, as his "secret word."
2. Play consists of alternate turns of naming a "test word," whose constraints are the same as on the secret words, and the opponent answering how close the test word is to his secret word.
3. Closeness is measured in jots; each jot is a one-to-one letter match, and independent of which word is the test word. GLASS versus SMILE or SISSY is 2 jots.
4. The first player to guess his opponent's secret word wins.

Constraints on a JOTTO program are:

First, it must have a dictionary of all possible words at the outset of each game. (The modification of adding newly experienced words to its dictionary is trivial in practice and not worth the programming effort, especially since one wants to avoid adding word-like typing errors, etc.) The (unacceptable) alternative is to have a letter-deducing algorithm and then a "word proposer" to order the 5 factorial = 120 combinations (perhaps based on digram frequencies and vowel constraints) once all 5 letters are found.

Second, the most use the program can make of the jots from a given test word is to eliminate from its list of "possible secret words of opponent" all those which do not have that number of jots against that test word. Hence, each test word should be chosen to maximize the expected number of words eliminated, i.e., maximize the expected information derived. If test word A divides the list of N possible secret words into n0 words of no jots, n1 of 1 jot, etc, and all words in the list are equally likely to be the opponent's real secret word, then we expect

$$-\sum_{i=0}^{5} \frac{n_i}{N} \log_2 \frac{n_i}{N}$$ bits of information

from giving test word A. (See Pierce, "Symbols, Signals, and Noise," page 84.) The best test word to give is the one (out of all those in the dictionary) which gives the highest number of bits expectation.

And third, since the program is to interact with humans, a certain amount of human engineering is desirable. For example, many humans feel cheated if the computer (or even another human) wins because its word is inordinately obscure; nevertheless, they do not want to be limited in their own choice of secret words. Thus CRWTH is fair game for humans, but not for computers!

A JOTTO program has existed for a couple of years at MIT's A.I. Lab, which program seems to have survived the test of time and thus

deserves documentation; this program will be the subject of the rest of this memo.

JOTTO is written in machine language (MIDAS) for the PDP-6/10, a computer whose instruction set is very symmetrical, complete, powerful and easy to think in. It takes about 1249 words of core, plus 3 words for each of about 7000 dictionary entries.

Since it employs the information-theory criterion described above for test word choice, its sequence of test words in guessing a given secret word is fixed. This allows pre-computing, storing and table look-up of the first few (usually three) test words. The game tree is stored to the depth where the list of possible words is 100 or less. For example, its first test word is always TEARS.

There are two second-order effects in choosing test words, neither of which is dealt with in any way in the program. The first regards permutation groups. Once all 5 letters are known, all one can do is sequence through all anagrams (the program does it randomly). Thus, if it takes several test words to narrow the field down to one particularly large permutation group, play might be substantially improved by some scheme such as weighting the information contribution of words exponentially as the size of their permutation group. This seems unnecessary in practice, however, since the large permutation groups are found fairly early in the game tree.

The other effect is somewhat subtler. It is conceivable that the test word with highest expectation at the current point in the game has a good chance of getting us to a point where we will NOT have any particularly good test words available. That is, it may be worth trading a fraction of a bit on this turn for a larger (on the average) fraction of a bit on, say, our next turn. I am indebted to Bill Gosper for pointing out this possibility; the computation required, however, is impractical, and besides, the program seems to do acceptably as is.

The previously mentioned dilemma regarding CRWTH is resolved by storing with each word one additional bit which says common/uncommon. The program never selects an uncommon word for its secret word. Since the dictionary is about 50/50, this amounts to giving the human an advantage of about 1 bit. This is not terribly large compared to the 2+ bits per test word the program gets; the program seems to hold its own anyway. Also the program never uses an uncommon word as a test word, which can also handicap it slightly, but one would not expect much improvement over the best test word among the 3500 common words. (This is borne out by best common versus best overall test word comparisons made in compiling the pre-stored opening test words.) An exception is when all 5 letters are known; then guessing is random among all permutations which are in its dictionary, irrespective of the common/uncommon bit. At this point it changes its phraseology from, "MY TEST WORD: TEARS" to, "IS YOUR WORD CRWTH?"

The dictionary was compiled from Webster's Seventh New Collegiate. A few mistakes have been found; the program doesn't know the word STUCK. These errors have been accumulated, and will (I hope) soon be corrected. Also aiding the editing of the dictionary are comparisons with Steve Brown's jotto dictionary at the MIT RLE PDP-1 (apparently mostly compiled from Webster's New World Dictionary, College Edition), and George A. Miller's National Institutes of Health jotto dictionary (compiled from Funk and Wagnall's unabridged).

Dictionary errors are ameliorated by the philosophy behind the program: polite I/O hiding powerful computation. It gives a one-line explanation when loaded, and at the beginning of each game asks, "WOULD YOU LIKE TO GO FIRST?" A Y or N is sufficient to answer any question it asks. When either player wins, it asks whether you want to continue guessing its secret word (or to have it continue guessing yours). It does not keep score or gloat over its wins. If you type a non-word at it, it asks, "ARE YOU SURE THAT'S A WORD?" and Y lets you cheat. (Or you can interpret the contents of location SECRET as sixbit and cheat that way!) Rubout deletes all of an incompletely-typed test word. Typing "?" when it's your turn to give a test word makes it ask, "WANT TO GIVE UP?"

One of the nicest features to add would be a back-up, so that when it says, "I GIVE UP. DO YOU WANT TO TRY TO GUESS MY WORD?" you can say, "Oops -- I meant to give you 3 jots on TEARS instead of 2 ... sorry." Because of the sequential word elimination, this would necessitate a small wait while the updated jot sequence was re-processed.

Another feature would be reversion to the letter-finding and permutation-guessing strategy instead of just giving up when it finds the list of possible secret words has shrunk to zero. This, however, is not worth the pain of programming it. Yet another "feature" would be to say, "I GIVE UP. WHAT WAS YOUR SECRET WORD? ----- YOU GAVE ME 2 JOTS ON TEARS AND THAT HAS 3 JOTS, GRIPE, GRIPE ..."

Each time a new game is started, ITS, the time-sharing system under which JOTTO runs, is called, and the binary values of the current date, time of day (both in units of 4.069 microseconds and in sixbit HHMMSS), and run time of this copy of the JOTTO program are XOR-ed together. This pseudo-random quantity is then used to select the program's secret word.

Programmers: please use or implement a feature to list words alphabetically down each column per page instead of across each row! The improvement in readability is at least 1000%.