# *Mnemonic Phrase Generation using Genetic Algorithms and Natural Language Processing*

James Mountstephens
School of Engineering and Information Technology
Universiti Malaysia Sabah
Kota Kinabalu, Sabah, Malaysia
james@ums.edu.my

*Abstract*— **Mnemonic phrases may help engineering students remember list information more easily. This paper describes the current progress of work to develop, implement and test two methods for the automatic generation of mnemonic phrases by computer. Techniques from artificial intelligence (AI) are drawn upon; more specifically from natural language processing (NLP) and genetic algorithms (GA). The first method approaches mnemonic phrase generation directly using NLP; the second approaches it as an optimisation problem to be solved by GA.**

***Keywords—learning; memory; mnemonics; genetic algorithms; natural language processing***

## I. INTRODUCTION

Memorisation is an integral part of learning in virtually all disciplines and, although the development of thinking skills and creativity are equally important, the training of professionals such as engineers depends on the transfer and long-term retention of large bodies of specialist knowledge. Such knowledge may come in different modalities ranging from the symbolic and conceptual, which are largely presented in text form, to the spatial and relational which are often presented graphically [1].

Unlike sentences, where the individual words interact to form an integrated meaning, *lists* are a form of unstructured text that may be especially hard to memorise since their items often have no obvious relationship to each other. Examples of lists in engineering might include colour coding for resistors [2], the activity sequence of metals [3], and compounds in order of solute and solvent structure [4].

For millenia, people have devised techniques to memorise information, known generally as mnemonics [1,5]. For the memorisation of lists, *mnemonic phrases* have often been used. The most common type of mnemonic phrase is a sentence whose words have the same initials as those in the list of words to be memorised. A mnemonic phrase has the same number of words as the original list but, as a sentence, with words chosen to give it the vivid and coherent overall meaning that is lacking in the original list, it is intended to be easier to remember.

Some examples of mnemonic phrases for different topics include the following. For the list of colours {red, orange, yellow, green, blue, indigo, violet} the mnemonic phrase 'Richard Of York Gave Battle In Vain' is often used. The eight planets {Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune} may be better memorised with the phrase 'My Very Excellent Mother Just Served Us Nachos'. Lastly, students of music often learn the notes {E, G, B, D, F} on the musical stave as 'Every Good Boy Deserves Food'.

Although not unequivocal, the concensus among studies testing the use of mnemonic phrases is that they have the potential to aid learning in many situations [6,7,8]. However, currently, the use of mnemonic phrases relies on their pre-existence or on the creativity of an individual learner in devising new phrases to suit their needs. Little work has been done on the automatic generation of mnemonic phrases by computer. Computer-generated mnemonics could be effortlessly available to all students, could potentially be applied to any kind of list information, regardless of discipline, and might also be extended to cover other types of knowledge.

This paper describes the current progress of work to develop, implement and test two methods for generating mnemonic phrases automatically. Techniques from artificial intelligence (AI) are drawn upon; more specifically from natural language processing (NLP) and genetic algorithms (GA). The first method approaches mnemonic phrase generation directly using NLP; the second approaches it as an optimisation problem to be solved by GA.

## II. METHOD

### *A. Overview*

For an input list *L* of *n* words with initials *I*, the output mnemonic phrase *MP* will also be a sequence of *n* words with initials *I*. The question is: which sequence of *n* words? It is useful to view the problem as one of selecting a sequence of *n* words from a master lexicon (word list) *W* that satisfy a number of conditions. *W* may be large and many sequences may be possible. We require the 'best' sequence of words and the conditions, described next, should reduce the possibilities to a single preferred sequence.

A precondition for the process is that the available words in *W* should be memorable in themselves. Words are not equally memorable or important; for example, content words such as nouns may be more memorable that utility words such as determiners and conjunctions. Psycholinguistic studies [9] have measured the imageability, concreteness, meaningfulness and familiarity of common nouns and it seems reasonable to restrict

978-1-4673-6355-6/13/$31.00 ©2013 IEEE 26-29 August 2013, Bali Dynasty Resort, Kuta, Indonesia

*W* to nouns with high values for these measures. Familiarity and importance may also be estimated using linguistic corpus analysis [10] to measure the frequency of occurrence of potential words in W and to select only those with high frequencies.

The most basic and least constrained *MP* would be any sequence of *n* words with the same initials as the input list. Eg. For the list of organic elements *L* = {carbon, hydrogen, nitrogen, oxygen, phosphorus, sulphur} the 'c' could be one of {cat, cloud, consequence…}, the 'h' could be one of {hat, harlequin, hot…} and so on. Clearly, a vast number of sequences $S_0$ are likely to exist and most would not be useful mnemonics or even genuine sentences; they might be expected to have no more memorable structure or content than the original list *L*.

Imposing grammatical constraints would eliminate a large number of sequences in $S_0$. The remaining sequences $S_1$ would be well-formed sentences and might be expected to be more memorable than an unstructured sequence of words. $S_1$ may still be very large since there may be many ways to assign parts of speech (POS) to the *n* words to form a grammatical sentence (eg. {article noun verb adj noun}, {article adj noun verb noun} etc.). Alternately, $\mathrm{S}_1$ may be empty if the word list *W* and grammatical rules do not allow assignment of POS for a particular set of initials *I*. For example, there are no prepositions with initials 'x' or 'z'. Two ways to satisfy grammatical constraints will be described in sections II.C and II.D. They form the heart of the method proposed here.

But even grammatical sentences need not be memorable. The sentence should also be *meaningful* if it is to be remembered. The sentences 'doughnuts belay pots' and 'dogs bite people' both have the same initials but the latter is clearly more meaningful as it expresses a fact that we can relate to. A meaning for the first sentence can be found but it requires effort since those words are not often found together and 'doughnuts' and 'pots' do not have the same semantic relationships as 'dogs' and 'bite' and 'people'. Ensuring or measuring the meaningfulness of sentences is a difficult and open research question. We will be pragmatic here and settle for measuring a kind of *coherence* between words in the sentence: are the words somehow related? Measures of semantic similarity between pairs of words have been developed [11] and may be used to evaluate full sentences. Coherence could also be measured by how likely the words in the sentence are to co-occur: a statistical measure that does not consider semantics directly [10].

With this overall approach to the problem in mind, some required linguistic background is given next, before the specifics of the two methods to achieve grammaticality are presented.

### *B. Grammar, Parsing and Generation*

The grammatical structure of sentences is a primary topic in NLP. Sentences may either be parsed or generated, processes that generally use a *grammar*, a set of rewrite rules for parts of speech (POS). An example of a very simple phrase structure grammar (PSG) [12] would be the following shown in Fig 1:

S → NP VP
NP → (r) (j) n (PP)
VP → v (NP) (PP)
PP → p NP

Fig. 1. Example Phrase Structure Grammar. Abbreviations used: sentence, noun phrase, verb phrase, article, adjective, noun, prepositional phrase, verb, preposition.

Parsing takes an existing sequence of words and attempts to assign a hierarchical set of POSs to each grammatical unit and, ultimately, to individual words. Using the phrase structure grammar above, the sentence 'dogs bite people' might be parsed as follows:

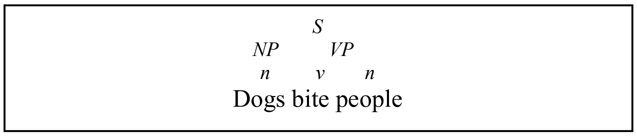

Fig. 2. Parsing the sentence 'dogs bite people' using the PSG in Fig 1.

Note that this simple grammar would not be able to parse the ungrammatical sequence of words 'run cat heretic' and would therefore be capable of distinguishing between grammatical and ungrammatical sentences. However, in general, there is no single deterministic grammar that represents English in all its complexity. Most modern parsers are probabilistic and will not accept or reject a sequence so strictly: a probability is given instead. Indeed, the question of detecting ungrammatical sentences itself is difficult and an active research topic [13].

Grammars may also be used to directly generate sentences which, due to the nature of their construction, are guaranteed to be grammatical. We now consider the application of grammars to the problem of generating mnemonic phrases but note that this should not be confused with the NLP topic of sentence generation which attempts to generate sentences to express content that is typically not already in sentence format.

### *C. Direct Grammaticality using Phrase Structure Grammar*

Given a list *L* of *n* items, any reasonable PSG should be able to generate a sequence of *n* POSs so that the sequence is a sentence. The added constraint here is that the *n* placeholders are bound to the initials *I*. PSG rules may be used recursively to divide the sequence up and to assign POS to the placeholders so that, in the end they each have a basic POS such as noun, verb or adjective. If the assignment is possible, it is guaranteed to be a grammatical sentence. An example of this process successfully applied to L={E, G, B, D, F} using the PSG from Fig 1 is shown below.

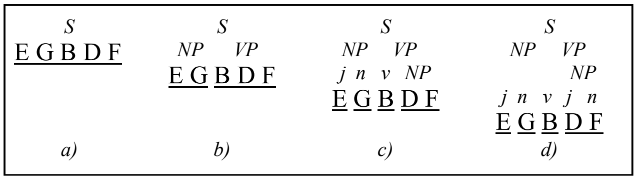

Fig. 3. Success in recursively dividing the initials and assigning POS using the PSG in Fig 1.

A selection of memorable and coherent words from W for these initials and parts of speech might yield the sentence ‘Evil Giants Boil Dirty Feet’, which is a reasonable mnemonic phrase.

The division of the word sequence may vary. In this example, the initial split could have been NP = {E,G,B} and VP = {D,F} instead. Different divisions will yield different sentence structures and in certain cases will produce a sentence structure for which words with the correct initial for the assigned POS do not exist.An example of this situation might occur for *L*={carbon, hydrogen, nitrogen, oxygen, phosphorus, sulphur} using the same PSG.

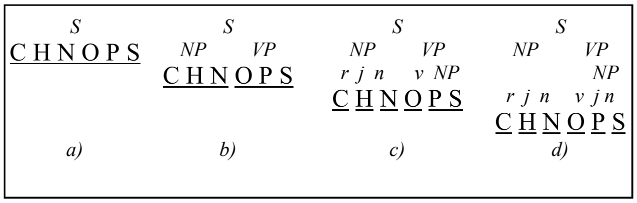

Fig. 4. Failure in recursively dividing the initials and assigning POS using the PSG in Fig 1. No article with initial ‘r’ exists.

At stage c), the initial ‘C’ has been assigned the POS ‘article’ but there is no English article beginning with ‘C’, rendering the sentence unrealisable. Steps must be taken to check if a generated sentence structure is compatible with the given initials and if not, to redivide the sequence and recurse again, which will assign a different POS to the initial in question.

Given that a sentence structure compatible with the given initials can be found, words must be selected to fill that structure. With no further contraints, any word in W with the correct POS and initial may be used in each position of the sequence, providing a potentially large set of possible sentences. But we have said that we require the words in the sentence to be meaningful or at least coherent so a measure of coherence for any given sequence must be used to select among the possibilities. Pairwise measures of semantic similarity [11] and of pairwise frequency of co-occurrence [10] can be used to estimate how well the particular sequence fits together as a whole. These measures are calculated for each possible pair of words in a given sentence and totalled. The sentence with the highest coherence is selected.

Examples of low vs. high coherence for *L*={carbon, hydrogen, nitrogen, oxygen, phosphorus, sulphur} might be: ‘cavernous hydrangeas need oral practical spinnets’ vs ‘captain haddock never opens private ships’. Words in the latter sentence are related and form a coherent meaning that requires effort to achieve in the former sentence.

### *D. Indirect Grammaticality using Genetic Algorithms*

In contrast to the approach of the previous section, one may also treat the task as an optimisation problem: generate a set of possible sequences of *n* words, calculate an objective function for each that measures grammaticality and choose the sequence with the highest value. Many methods for optimisation exist but not all can be applied to a problem where the objective function is analytically unknown, as is the case here.

Genetic Algorithms (GAs) are able to solve a wide variety of optimisation problems [14] and well-suited to the task at hand. Taking inspiration from biological reproduction and evolution, the basic unit in a GA is the chromosome, which comprises a sequence of genes taking particular values known as alleles. For the generation of a mnemonic phrase for {carbon, hydrogen, nitrogen, oxygen, phosphorus, sulphur}, example chromosomes might be as below:

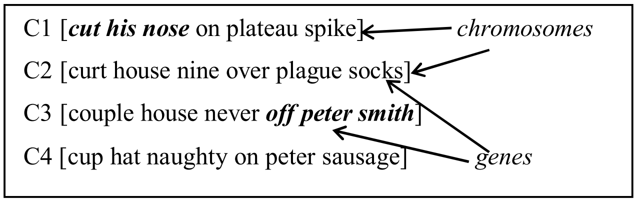
C1 [***cut his nose*** on plateau spike]
C2 [curt house nine over plague socks]
C3 [couple house never ***off peter smith***]
C4 [cup hat naughty on peter sausage]
*chromosomes*
*genes*

Fig. 5. Four example chromosomes, each with with six genes.

Although the specifics of GAs are beyond the scope of this paper, the essential idea is that an initial population of chromosomes are assessed for fitness (the GA version of an objective function) and the fittest allowed to reproduce to form new chromosomes. In the example above, the left and right halves of C1 and C3 have high fitness and might combine to form the better chromosome [cut his nose off peter smith] which is a grammatical sentence and a reasonable mnemonic phrase for the list of organic elements. The process of evolution is continued until chromosomes with a desired level of fitness are produced.

To apply GAs to the task at hand, a fitness function that evaluates the grammaticality of a given sequence of words must be found. It is the novel contention in this paper that parsers may be used as a fitness function for sentences. During evolution, each chromosome will be parsed by a general English parser that can give it a high or low rating for grammaticality and the fittest will reproduce to yield fitter offspring and hopefully genuine sentences.

Since we require coherence among words in a sentence, the fitness function will also use the pairwise measures of semantic similarity and frequency of co-occurrence described in section II.C.

## III. IMPLEMENTATION & TESTING

A Java implementation of the methods just described is almost complete and will be tested on engineering undergraduates soon. A considerable body of custom code has been developed and a number of existing resources are also used.

The direct method for achieving grammaticality described in II.C has been coded entirely from scratch. Input lists of initials are recursively divided and assigned parts of speech. Each division and assignment is checked to see if the assigned parts of speech exist for the initials in the current division. If not, a different division is generated and the process repeated. If this is still unsuccessful, the previous level in the hierarchy is returned to and different divisions are tried there. The code is able to flexibly use different phrase structure grammars.

The indirect method for achieving grammaticality described in II.D employs the following existing libraries for sentence parsing and running genetic algorithms.

- JGAP Java Genetic Algorithms Library [15]. JGAP is a freely-available implementation of many established genetic algorithms. It offers classes and methods for generating and evolving populations and calculating fitness functions.
- Stanford Parser [16]. The Stanford Parser is a popular English parser that is freely-available and well-documented. It is used here to parse a chromosome of words and return a probability taken here to reflect the grammaticality of the chromosome. This value is used in the fitness function of JGAP.

The precondition of using meaningful words W is currently being met with use of the MRC Psycholinguistic Database [17], which is machine-readable dictionary of 150837 words, of which about 2500 are annotated with up to 26 linguistic and psycholinguistic attributes including concreteness, meaningfulness, imageability and familiarity. Words with high measures of these are selected.

Finally, the estimation of sentence coherence uses a combination of the following resources:

- British National Corpus XML edition [18]. The BNC is a 100-million word English corpus, with texts sampled to be well-representative of the language as a whole. It is used here to calculate frequencies of co-occurrence between pairs of words which can help to distinguish between normal and random sequences of words.
- WS4J Semantic Similarity Library in Java [19]. This library implements eight established measures for the semantic similarity of pairs of words. Like the BNC it is used to estimate the relatedness or disparity of a sequence of words.

## IV. CONCLUSION

This paper has described progress on the development of two methods for the automatic generation of mnemonic phrases for list information. The methods are conceptually well-developed and almost fully-implemented. Testing will be carried out and reported on soon.

There is much scope to extend the basic methods given here. In particular, mnemonic phrases could be directed towards the type of content to be memorised or personalised to the background and preferences of individual students to make them more memorable. For example, to personalise a mnemonic phrase for the list $L$={carbon, hydrogen, nitrogen, oxygen, phosphorus, sulphur} towards asian learners, words more likely to be memorable might be used: ‘Come Home, Neighbour Of President Sukarno’. To personalise the same list for a Briton to remember, an alternative formulation might be more memorable: ‘Chelsea Has No Other Players, Sorry!’

## REFERENCES

[1] A. Searleman and D. Herrmann, Memory From A Broader Perspective, New York: McGraw-Hill, 1994.

[2] P.R. Clement and W.C. Johnson, Electrical Engineering Science. McGraw-Hill. p. 115. 1960.

[3] W.L. Jolly, Modern Inorganic Chemistry (2nd ed.), New York: McGraw-Hill, 1991.

[4] E.V. Anslyn and D.A. Dougherty, Modern Physical Organic Chemistry. University Science Books, 2006.

[5] S. Glynn, T. Koballa, D. Coleman, “Mnemonic Methods”, Science Teacher 70(9), 2003, pp. 52-55.

[6] J.R. Levin, M.B. Nordwall, "Mnemonic vocabulary instruction: Additional effectiveness evidence", Contemporary Educational Psychology 17 (2), 1992, pp. 156–174.

[7] S.S. Seay, "The Use/Application of Mnemonics As a Pedagogical Tool in Auditing", Academy of Educational Leadership Journal 14 (2), 2010.

[8] R. O'Hara, et al,"Long-term effects of mnemonic training in community-dwelling older adults", JPsychiatrRes 41(7), 2007, 585-90

[9] K.J. Gilhooly and R.H. Logie “Age of acquisition, imagery, concreteness, familiarity and ambiguity measures for 1944 words”, Behaviour Research Methods and Instrumentation 12, 1980, pp.395-427.

[10] John Sinclair, Corpus, concordance, collocation, Oxford University Press, 1991.

[11] A. Budanitsky, G. Hirst, "Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures". Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics. Pittsburgh, 2001.

[12] R. Borsley, Syntactic theory: A unified approach. London: Edward Arnold, 1991.

[13] J. Wagner, J. Foster, J. van Genabith, “Judging grammaticality: experiments in sentence classification”. CALICO Journal, 26 (3). 2009, pp. 474-490.

[14] M. Mitchell, An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.

[15] http://jgap.sourceforge.net/

[16] http://nlp.stanford.edu/software/lex-parser.shtml

[17] M.D. Wilson, “The MRC Psycholinguistic Database: Machine Readable Dictionary, Version 2”. Behavioural Research Methods, Instruments and Computers, 20(1), 1992, 6-11.

[18] L. Burnard, G. Aston, The BNC handbook: exploring the British National Corpus. Edinburgh: Edinburgh University Press. 1998.

[19] http://code.google.com/p/ws4j/