Neurocognitive Approach to Creativity in the Domain of Word-Invention

Maciej Pilichowski[1] and Włodzisław Duch[2]

[1] Faculty of Mathematics and Computer Science
[2] Department of Informatics, Nicolaus Copernicus University, Toruń, Poland
macias@mat.umk.pl, Google: W. Duch

Abstract. One of the simplest creative act is the invention of a new word that captures some characteristics of objects or processes, for example industrial or software products, activity of companies, or the main topic of web pages. Neurocognitive mechanisms responsible for this process are partially understood and in a simplified form may be implemented in a computational model. An algorithm based on such inspirations is described and tested, generating many interesting novel names associated with a set of keywords.

1 Introduction

Creativity understood as "the capacity to create a solution that is both novel and appropriate" [1], manifests itself not only in creation of novel theories or inventions, but permeates our everyday life, including comprehension and production of text and speech. The best chance to capture creativity in computational algorithms is to follow neurocognitive inspirations [2,3].

Despite the limitation of the current knowledge of the neural processes that give rise to the higher cognitive processes in the brain it is possible to propose a testable, neurocognitive model of creative processes. A review of the psychological perspectives and neuroscientific experiments on creativity has already been presented in [2,3]. Three elements are essential for computational model of creativity: first, a neural space for encoding probabilities of strings of letters and phonemes in a given language, leading to automatic associations of keywords with semantically and phonetically related words through spreading of neural activation; second, imagination based on combinations of phonemes or letters, with probability that depends on the priming effects [4]; and third, the filtering of interesting results based on density of phonological and semantic associations. Creative brains are well trained, have greater imagination, combine faster basic primitives, and pick up interesting combinations of these primitives through emotional and associative filtering. For novel words filters based on phonological and semantic plausibility may be proposed. Phonological filters may be realized using autocorrelation matrix memories or probabilistic models. Semantic filters should evaluate how many words have similar meaning to the new word, including similarity to morphemes that the words may be decomposed to.

The Mambo algorithm described here is one of many possible models based on these principles. It is tested on the invention of novel, interesting words that are suitable as names for products, companies, web sites or names of various entities.

2 Algorithmic Considerations

Initially the Mambo system does not have any *a priori* linguistic knowledge. Google Web 1T 5-gram [5] dictionary is used as the training source. This dictionary is based only on word frequency, therefore it is necessary to perform at least minimal spell-checking before using this data for training. The LRAGR [6] and SCOWL [7] dictionaries have been used to correct the misspelled words. An unexpected and quite surprising problem is the relative shallowness of such large dictionary provided by Google – over 40% of words that exist in standard English dictionaries are not yet present in the Google dictionary, probably due to the large number of rare and outdated words in English [8].

Dictionary presented to the system is used to learn morphological regularities using probabilistic model. This model is somewhat easier to control than the autocorrelation matrix memory that may also be used for such learning. Word frequencies are used in the learning process. However, using raw word frequencies would lead to very frequent presentation of some common words (mostly articles and conjunctions). For example "the" occurred 23 billion times, while "of", the next most common word, occurred 13 billion times only. On the other end of the list such word as "viscoses" has frequency around 200. To avoid these huge differences logarithmic scale for frequencies is used.

Filtered data (words and their frequencies) are the source of knowledge for the system – depending on selected symbol set, data can be presented as letters ("the" $\rightarrow$ "t", "h", "e"), phonemes ("the" $\rightarrow$ "ð", "ə"), in a semi-letter form ("the" $\rightarrow$ "th", "e"), only for English, or in a mixed form (a word is processed in several forms at the same time). In our implementation mixed mode with letters and phonemes causes huge CPU load because conversion to a phoneme representation is done via a slow external program. The pure phoneme form, written using the IPA (International Phonetic Alphabet) codes, is hard to read for most people. The semi-letter form is easily readable and as fast to process as the letter one, allowing to keep the atomicity of most characteristic phonemes ("sh", "ch", "th", "ph", "wh", "rh").

Semantics. The system learns the structure of the words but not their meanings. Thus it does not have such flexible associations as people. For example, "light" may mean "small weight", but it may also be associated with "daylight". The program is unaware of intended or unintended collocations of symbol strings. Advertising some product it is helpful to associate "possibilities" with "great" (positive association), rather than "problems" (negative association). Currently the Mambo system does not include word valence, but such an extension is not difficult, as affective word lists are readily available. Without it in real applications better results are obtained if a distinct sets of priming words with only negative associations ("bound", "less"), or words with only positive associations ("more", "energy"), are provided, but not both sets.

Associations. In a typical situation a description of the product will be given, read, and names for this product will be proposed by a human expert. Reading such description leads to priming of the network, making it more susceptible to activation for the same and related words. Priming is one of the effects of implicit memory which increases the probability of recalling the previously encountered item, not necessarily in the same form (as defined by Kalat [9]). Priming is achieved by repetition of the presented item or by presenting items associated with the target [4].

It is not easy to implement realistic priming in an automatic way. WordNet database [10] that collects words with similar meanings into "synsets" may help, but despite extensive efforts the content of WordNet is still far from perfect, and even for simple queries questionable associations arise. Therefore WordNet is used only to help the user to prepare the priming set.

Word determinant (prefix or suffix) is defined as the shortest prefix (suffix) of word which is not a prefix (suffix) of any other word in the dictionary, with the exception of derivatives of this word. For example, the prefix determinant of "education" is "educat" ("educated", "educational", and other forms are derivates of "educate"). Possible backward associations may become a problem: the system can create a word which is relevant to the priming set, but this word may be related in a much stronger way to words outside of the priming set. As an example consider the priming item "owl", and the system's association with more frequent "bowl" that may dominate [11] inhibit correct associations to "owl". There is no good mechanism that could control such hijacking, the system can only enforce existence of word determinants from the priming set. This enforcement has diminished the problem but did not eliminate it completely. Restricting created words to those that contain at least one word determinant will assure that results will be associated with the desired (priming) word set.

Originality. Words created by the system should be original, therefore copies of previously learnt words should be avoided. At the filtering level all shortened forms of already known words are rejected ("borrow" $\rightarrow$ "borr") and words with superficial changes ("borrow" $\rightarrow$ "borrom"). At first this looks like an obvious decision, but for real world application it may be too strict – just consider "sorrow", "barrow", or "burrow". Originality plays the most important role in applications of Mambo system. Brand names or company names should distinguish themselves from those already existing (although similarity has some advantages, for example several "Imtel" or "Indel" companies exist). The same rules are also used in order to avoid numerous variations on a single created word.

Imitations. Compound words, such as the "bodyguard", "brainstorm" or "airmail", are highly ranked by the Mambo system. While testing the system on concepts related to aviation quite compelling word "jetmail" has been created, but only because the system was primed with the word "jet" and there was word "• • •mail" in the dictionary. The first part gave it a high association rank, the latter contributed to good linguistic quality of the result. Thus creation of compound words in the Mambo system may lead to a form of plagiarism – the system is creative only if humans were creative before. Although this may be corrected in the final stage (searching the Internet to determine if the word is new), for the purpose of testing all compound words have been removed from the dictionary to guarantee that words created by the Mambo system result from its internal algorithm, and not from a simple word takeover.

3 Word Rank Function

Words are ranked by their linguistic (phonological) and semantic quality, depending on associations that they may invoke. The word rank function $Q'(w)$ is defined as:

$$Q'(w) = \prod_{i=0}^{|ng(T(w))|-1} q(ng(T(w))[i]) \tag{1}$$

where function q is a dictionary function, for a given ngram returning numerical value based on previously processed dictionary data (see below), $T(w)$ is a composition of word w transformations, ng is a function partitioning symbols in w into overlapping ngrams of length $N_{ng}+1$, shifted by S_{ng}. Function $ng(w)$ returns a sequence of strings (ngrams):

$$\begin{aligned} ng(w) = \langle & w[0 : N_{ng}], w[S_{ng} : S_{ng} + N_{ng}], w[2S_{ng} : 2S_{ng} + N_{ng}], ..., \\ & w[nS_{ng} : nS_{ng} + N_{ng}]\rangle \end{aligned} \tag{2}$$

where $w[i : j]$ represents string of symbols at positions i to j in the word w, and $nS_{ng} = |w| - N_{ng} - 1$. In most cases these values are set to $N_{ng} = 2$ and $S_{ng} = 1$; for example, partition of ”world” is then ”wor”, ”orl”, ”rld”. Let $a \cdot b$ mean concatenation of sequences a and b, and let Σ be the alphabet. Some examples (for $N_{ng} = 2$, $S_{ng} = 1$) of word strings transformations are:

neutral transformation	$w \rightarrow w$	e.g. world $\rightarrow$ world
cyclic transformation	$w \rightarrow w \cdot w[0 : N_{ng} - 1]$	e.g. world $\rightarrow$ worldwo
mirror transformation	$w \rightarrow w[\|w\| - 1] \cdot w[\|w\| - 2] \cdot ... \cdot w[0]$	e.g. world $\rightarrow$ dlrow

Let us define also functions $h(w) = w[0 : |w| - 2]$, $t(w) = w[|w| - 1]$ and function $r(i)$. When $r(i) = i$ positional ngrams are used (ngrams which „remember” their position within a sequence), and when $r(i) = 0$ free ngrams are used.

Several transformations may be applied to words simultaneously. A model of such computation is defined by several user-defined parameters: function $r(\cdot)$, ngram parameters N_{ng}, S_{ng}, composition of transformations (T), the alphabet (Σ), and the exponent W estimating importance of the function $Q'(w)$ for a given model. The total word rank function is a product over models:

$$Q(w) = \prod_{k=1}^{\#models} Q'_k(w)^{W_k} \tag{3}$$

At least one model is always required; if there are more models specified the first one is assumed to be the main model, and the remaining ones are auxiliary. Only words ranked above some threshold will be accepted as “interesting”.

Function q is calculated using information from the dictionary of words and their frequencies, and optionally from a sub-dictionary containing the priming set Pr, which also contains words and their counts. Its purpose is to obtain probability-like measure of ngrams for words found in the training dictionary. For each specific model of creating words, data are put into distinct pairs of C and D tables. The steps to create these tables are presented below in the pseudo-C++ code; the calculations in lines marked with † symbol are dependent on the parameters passed to the system.

1. Saving data from $Dict$ to table D_d
```
   for each word w in Dict
     for each i-th ngram g in ng(T(w))
†      D_d[r(i),h(g),t(g)] += count(w)
```
2. Saving data from *Pr* to table D_p
```
   for each word w in Pr
     for each i-th ngram g in ng(T(w))
†      D_p[r(i),h(g),t(g)] = 1
```
3. Rescaling values
```
   for each element e in D_d
†  e = log_10(e)
```
4. Normalization of D_d, creating table C_d
```
   for b = 0 to depth(D_d)-1, step +1
     from y = 0 to rows(D_d)-1, step +1
†      C_d[b,y] = sum(D_d[b,y]) / sum(D_d[b])
       s = sum(D_d[b,y])
       for x = 0 to cols(D_d)-1, step +1
†        D_d[b,y,x] /= s
```
5. Creating table C_p
```
   for b = 0 to depth(D_p)-1, step +1
     for y = 0 to rows(D_p)-1, step +1
†      C_p[b,y] = sum(D_p[b,y]) > 0 ? 1 : 0
```
6. Adding table values
```
†  C = C_d + C_p
†  D = D_d + C_p
```
7. Normalization correction of the final tables
```
   for b = 0 to depth(D)-1, step +1
     for y = 0 to rows(D)-1, step +1
†      C[b,y] /= sum(C[b])
       s = sum(D[b,y])
       for x = 0 to cols(D)-1, step +1
†        D[b,y,x] /= s
```
8. Weighting the tables
```
   for each element e in C
†  e = e^W
   for each element e in D
†  e = e^W
```

The C and D tables serve to construct functions q and q' that may be interpreted as the probability of occurrence of a given ngram. For the first ngram h the function q may be identified as unconditional probability, and for the remaining symbols as conditional probability. In simplified notation $P(a_i) = P(a|i)$, where i is the position of a in word w over alphabet Σ. For subword v of word w let's also define:

$$P(b|a_i) = P(b_{i+|a|}|a_i); \quad h(v) = a, t(v) = b \tag{4}$$

Probability function P is calculated by a call to the table C and D: $P(a_i) = C[r(i), a]$, and $P(b|a_i) = D[r(i), a, b]$. Function $q(G)$, for a transformed sequence $G = ng(w)$, is defined as a product:

$$q(G) = P(h(G[0])_0) \prod_{i=0}^{|G|-1} P(t(G[i])|h(G[i])_i) \tag{5}$$

For the main model and positional ngrams the sequence G does not have to be treated as a continuous entity – it can be divided into independent sub-sequences in such a way that $G_0 \cdot G_1 \cdot \ldots \cdot G_J = G$. With the helper function that ranks ngrams:

$$qj(G', d) = P(h(G'[0])_d) \prod_{i=0}^{|G'|-1} P(t(G'[i])|h(G'[i])_{d+i}) \tag{6}$$

where G' is sub-sequence of G and d is initial depth of search in tables C and D, the counterpart of q for non-continuous case—function q'—is defined as:

$$q'(G) = \prod_{i=0}^{J} qj(G_i, d_i); \quad 0 \leqslant d_i \leqslant \mathrm{depth}(D) - |G_i|, \quad d_0 = 0 \tag{7}$$

Parameter J, defined by the user, controls the depth of search in matrix D for ngrams that are combined; when $J = 0$ there is no G division and $q' \equiv q$. Computationally the algorithm for functions q and q' seems to be straightforward – it is just some multiplication of values taken from the tables. However, when creating all words of a given length L the naive approach will not be acceptable due to the time complexity: for function q it is $O(|\Sigma|^L)$. To avoid lengthy computations optimized algorithm has been used, with cut-offs for words that fall below desired rank threshold at an early stage. Detailed discussion of these technical improvements are beyond the scope of this paper (Pilichowski, PhD thesis, 2009).

4 Results

The first test aims at finding a better name for Kindle electronic book reader that Amazon is advertising. Manually creating the whole priming set would be too tedious, therefore the user has to provide only the core set and the exclusion list (both in form of regular expressions). The priming set is obtained automatically from the dictionary adding the words that match regular expressions in the core set and do not match any words from the exclusion list.

The core priming set in this experiment was (each row is a concept group):

1.	acquir, collect, gather
2.	air, light$, lighter, lightest, paper, pocket, portable
3.	anyplace, anytime, anywhere, cable, detach, global, globe, go$, went, gone, going, goes, goer, journey, move, moving, network, remote, road$, roads$, travel, wire, world
4.	book, data, informati, knowledge, librar, memor, news, word$, words$
5.	comfort, easi, easy, gentl, human, natural, personal
6.	computer, electronic
7.	discover, educat, learn, read$, reads, reading, explor

The exclusion list contains: aird, airin, airs, bookie, collectic, collectiv, globali, globed, papere, papering, pocketf, travelog. The top-ranked results are below, with the number of domains that use the word, identified using Google at the end of 2008, given in the last column:

Created word	**Google word count**	**No. domains**
librazone	968	1
inforizine	–	–
librable	188	–
bookists	216	–
inforld	30	–
newsests	3	–
memorld	78	1
goinews	31	–
infooks	81,200	7
libravel	972	–
rearnews	8	–
informated	18,900,000	8
booktion	49	–
inforion	7,850	61
newravel	7	–
datnews	51,500	20
infonews	1,380,000	20
lighbooks	1	–
journics	763	1

Mambo has created several interesting new words, finding may words that are already used and have been created by people. The system has also proposed words the humans so far have used almost by accident – like “inforld”.

Another experiment was carried out to find the name for itself, i.e. the Mambo system, as it was preliminarily called. The parameters of all runs were exactly the same as above. The core priming set is presented in the table below:

1.	articula, name
2.	create, creating, creativ, generat, conceiv, build, make, construct, cook, formula, prepar, produc
3.	explor, discov, new$, newer$, newest$, newly$, imagin
4.	mean$, meanin, associat, idea$, ideas, cognitiv, think, thought, semant, connect, art$, artist, brain, mind, cogit
5.	system$, systems$, program, automat, computer, artifici
6.	wit$, wits$, witty$, smart, intell
7.	word, letter, languag

with the exclusion of one word: cookie. It is worth noting that this algorithm selected much fewer words as interesting than resulted from our previous experiments [3]; it has generated the following words:

Created word	Google word count	No. domains
semaker	903	9
braingene	45	–
assocink	3	–
thinguage	4,630	–
systemake	4	–
newthink	8,960	46
thinknew	3,300	43
assocnew	58	–
artistnew	1,590	1
semantion	693	6

Mambo system was capable of creating such words as “braingene” or “thinguage”. Fine-tuning the system (parameters, priming) may provide a lot of new words.

5 Conclusions

Inventing novel names is clearly a creative process that can be simulated using computational algorithms, and tested against human ingenuity. Many companies offer naming services, for example: Brands and Tags, Brighter Naming, Names and Brands, Named at Last, NameSharks. Models of neurocognitive processes involved in the process of creating and understanding novel words, capturing details at different levels of approximation, are worth developing. In this paper probabilistic approach has been presented. Despite its simplicity it gives quite interesting results. General neurocognitive principles of creativity are probably common to various tasks [2,3]. There is no doubt that algorithms based on neurocognitive inspirations are going to be quite useful in many interesting applications.

References

1. Sternberg, R.J. (ed.): Handbook of Human Creativity. Cambridge University Press, Cambridge (1998)
2. Duch, W.: Computational Creativity, World Congress on Computational Intelligence, Vancouver, Canada, pp. 1162–1169. IEEE Press, Los Alamitos (2006)
3. Duch, W., Pilichowski, M.: Experiments with computational creativity. Neural Information Processing – Letters and Reviews 11(4-6), 123–133 (2007)
4. McNamara, T.P.: Semantic Priming. Perspectives from Memory and Word Recognition. Psychology Press (2005)
5. Brants, T., Franz, A.: Web 1T 5-gram Version 1, `http://www.ldc.upenn.edu/Catalog/`
6. The Lexical Systems Group, The Specialist Lexicon, `http://lexsrv3.nlm.nih.gov/LexSysGroup/Projects/lexicon/current/`

7. Atkinson, K.: Spell Checker Oriented Word Lists, `http://wordlist.sourceforge.net/`
8. Cole, C: The Borgmann Project: Listing all the Words in English, `http://pl.youtube.com/watch?v=kU_CiErwkFA`
9. Kalat, J.W.: Biological psychology, 9th edn. Wadsworth Publishing (2006)
10. Miller, G.A., Fellbaum, C., Tengi, R., Wakefield, P., Langone, H., Haskell, B.R.: A lexical database for the English language, `http://wordnet.princeton.edu/`
11. Bowers, J.S., Davis, C.J., Hanley, D.A.: Automatic semantic activation of embedded words: Is there a "hat" in "that"? Journal of Memory and Language 52, 131–143 (2005)